Fabrizio Lombardi
Jing Huang
Editors

DESIGN AND TEST OF DIGITAL CIRCUITS BY QUANTUM-DOT CELLULAR AUTOMATA

Design and Test of Digital Circuits by Quantum-Dot Cellular Automata

Jing Huang, Fabrizio Lombardi

Northeastern University
Department of Electrical and Computer Engineering
360 Huntington Av.
Boston, MA, 02115

September 25, 2007

Contents

Preface xiii

Chapter 1 Introduction 1
1.1 Challenges 2
1.2 Previous Work 3
1.3 Contributions 4
1.4 Book Outline 7
References 8

Chapter 2 Nano Devices and Architectures Overview 11
2.1 Nanoelectronic Devices 12
2.1.1 Carbon Nanotube-based Devices 12
2.1.2 Nanowires 14
2.1.3 Molecular Electronic Devices 15
2.1.4 Single-Electron Devices 17
2.1.5 Resonant Tunneling Diodes 21
2.1.6 Spin Transistors 22
2.2 Nano-scale Crossbars 23
2.3 Architectures 25
2.3.1 SET Architecture 26
2.3.2 RTD Architecture 26
2.3.3 NanoFabrics Architecture 27
2.3.4 NanoPLA 29
References 33

Chapter 3 QCA 37
3.1 QCA Implementation 42
3.1.1 Metal QCA 42
3.1.2 Molecular QCA 44
3.1.3 Magnetic QCA 45
3.2 Clocking 46
3.3 Molecular Attachment 49
3.4 Power Gain and Dissipation 51
3.5 QCA Simulators 53
3.5.1 QCADesigner 54
3.6 QCA Circuits 56
3.7 Comparison of Nanotechnology Devices 61
References 64

Chapter 4 QCA Combinational Logic Design 69
4.1 Gate-based Combinational Logic Design 69
4.1.1 Gate-based Design of QCA with Existing Commercial Synthesis Tools 71
4.2 Logic Synthesis 73
4.2.1 AND/OR-based Logic Synthesis 73
4.2.2 Muroga's MV-based Logic Synthesis 75
4.2.3 MAjority Logic Synthesizer (MALS) 75
4.3 Structural Design 75
4.4 AND-OR-Inverter (AOI) Gate 76
4.4.1 AOI Gate Characterization 76
4.4.2 Defect Characterization of the AOI Gate 78
4.4.3 Logic Synthesis Using the AOI Gate 82
4.4.4 Conclusion 87
References 89

Chapter 5 Logic-Level Testing and Defect Characterization 91
5.1 Logic-Level Testing 91
5.1.1 Stuck-at Test Properties of MV-based Circuits 92
5.1.2 Test Set for MVs 95
5.1.3 C-Testability of MV-based Designs 96
5.2 Defect Characterization of Devices 99
5.2.1 Simulation Engines 101
5.2.2 MV Defect Analysis 102
5.2.3 Interconnect Defect Analysis 107

5.2.4 Probabilistic Analysis and Testing 111
5.2.5 Defect Analysis and Testing of QCA Circuits 116
5.2.6 Scaling in the Presence of Defects 133
5.2.7 Conclusion 140
References 141

Chapter 6 Two-Dimensional Schemes for Clocking/Timing of QCA Circuits 143
6.1 Clocking Analysis 144
6.2 Two-Dimensional QCA Clocking 146
6.3 Two-Dimensional Wave QCA Clocking 151
6.4 Examples of QCA Circuits 156
6.5 Feedback Paths 159
6.6 Simulation Results 160
6.6.1 2-to-1 Multiplexer 161
6.6.2 One-bit Full Adder 161
6.6.3 RS Flip-flop 161
6.7 Conclusion 162
References 168

Chapter 7 Tile-Based QCA Design 171
7.1 QCA Design by Tiling 174
7.2 Fully Populated Grid Analysis 176
7.3 Tiles Based on $\mathbf{3} \times \mathbf{3}$ Grids 179
7.3.1 Orthogonal Tile 179
7.3.2 Double Fan-out Tile 183
7.3.3 Baseline Tile 187
7.3.4 Fan-in Tile 190
7.3.5 Triple Fan-out Tile 192
7.4 Analysis of Results 195
7.4.1 Configuration Selection 196
7.5 Logic Analysis 196
7.6 Examples of QCA Circuits 200
7.6.1 One-bit Full Adder 200
7.6.2 Parity Checker 201
7.6.3 2-to-4 Decoder 206
7.6.4 2-to-1 MUX 208
7.7 Conclusion 210
References 211

Chapter 8 Sequential Circuit Design in QCA 213
8.1 RS Flip-flop and D Flip-flop in QCA 214
8.1.1 Defect Characterization of RS Flip-flop 216
8.2 Timing Constraints in QCA Sequential Design 219
8.2.1 Timing Constraints Using RS Flip-flops 220
8.2.2 Timing Constraints using D Flip-flops 221
8.3 Algorithm for Clocking Zone Assignment 221
8.3.1 Algorithm Outline 221
8.3.2 Algorithm Detail 223
8.3.3 Algorithm for Coplanar Device 226
8.3.4 Examples of QCA Circuits 227
8.4 Defect Characterization of QCA Sequential Circuits 229
8.5 Discussion and Conclusion 239
References 246

Chapter 9 QCA Memory 247
9.1 Introduction 247
9.2 Review of QCA Memories 249
9.3 Parallel Memory Architecture 252
9.3.1 Proposed Parallel QCA Memory Design 252
9.3.2 Clocking Considerations 255
9.3.3 Discussion and Comparison 257
9.3.4 Simulations 261
9.4 Serial Memory Architecture 263
9.4.1 Memory Design by Tiling 263
9.4.2 Clocking and Timing 266
9.4.3 QCA Tiles 268
9.4.4 Simulation 271
9.4.5 Conclusion 285
References 285

Chapter 10 Implementing Universal Logic in QCA 287
10.1 Universal Gate 288
10.2 Universal Gate Designs 289
10.2.1 AND/OR-based Synthesis 290
10.2.2 MV-based Synthesis 290
10.3 Memory-based LUT 294
10.4 Multiplexer-based LUT 298
10.5 Discussion and Conclusion 301

References 302

Chapter 11 QCA Model for Computing and Energy Analysis 305
11.1 Review on Reversible Computing 306
11.2 Mechanical Model 308
11.2.1 Model of QCA Cell 309
11.2.2 Steady State Energy of QCA Devices 312
11.3 Entropy and Dissipation Analysis 315
11.3.1 Operation of the Mechanical Cell 315
11.4 Landauer and Bennett Clocking Schemes 320
11.5 Conclusion 323
References 325

Chapter 12 Fault Tolerance of Reversible QCA Circuits 327
12.1 Hardware Redundancy Techniques 328
12.2 Majority Multiplexing in QCA 333
12.2.1 Fault Tolerant Capacity 334
12.2.2 Restoration Speed of Multiplexing 336
12.2.3 Summary 338
12.3 Reversible Computing and Fault Tolerance 339
12.4 Energy Dissipation of a Reversible MV Multiplexing System 341
12.4.1 System Without Fault 341
12.4.2 Dissipation in Fault Correction 342
12.5 Conclusion 344
References 347

Chapter 13 Conclusion and Future Work 349

App. A Preliminary for QCA Mechanical Model 353
References 356

App. B Validation of Mechanical Model 357
B.1 Validation of Static Energy Analysis 357
B.2 Validation of Dissipation Analysis 358
References 360

App. C Energy Dissipation Analysis of Circuit Units 363

About the Authors 367

Preface

Emerging technologies have been a topic of great interest over the last few years; as predicted by the Technology Roadmap of the Semiconductor Industry, CMOS as today's dominant technology for manufacturing computer systems by Very Large Scale Integration (VLSI) will be encountering serious hurdles in the future. The projected expectations in terms of device density, power dissipation and performance necessitate radically different technologies that provide innovative solutions to integration as well as computing. So-called emerging technologies have been advocated from disparate sources (both industry and academia) to meet these ambitious objectives, while realizing the ever-higher demands posed by the ubiquitous nature of computing in modern society.

This book addresses one of the most interesting among emerging technologies for digital design, Quantum-dot Cellular Automata (QCA). Over the last few decades since its inception at the University of Notre Dame, QCA has dramatically evolved in a dynamic and exciting field of investigation with contributors from all over the world. QCA is a challenging technology that due to its unique structural and operational features represents a revolutionary departure from current practice. QCA relies on principles that are fundamentally different from CMOS and therefore, it may offer unprecedented advantages to solve those challenges that are expected to occur at the end of the technology roadmap. For example, as its operation is based on Coulombic interactions, designers of QCA-based circuits must be made aware of the implications that selective properties (such as those based on switching and clocking) may come into play once a QCA circuit is embedded on a planar layout.

Numerous journal and conference articles have appeared in the technical literature; the last few years have also seen an increased number of professional meetings in which many sessions have been devoted to advances in QCA. However, QCA necessitates an understanding of physical and electrical phenomena that are not readily available from a single source. This book provide a focused reference by which up-to-date topics are treated in detail with direct impact on research

and practical implementations; moreover, its contents reflect an interdisciplinary approach by which scientists and engineers can mutually benefit. Only essential mathematics and physics are presented, while devoting substantial coverage to design and manufacturing issues as well as related topics such as testing, defect modeling and performance.

In this book, we have combined topics that cover the whole spectrum of interests in QCA: starting from a basic characterization at device-level, circuits and modular digital systems (such as memories and universal logic) are introduced to the reader within a systematic and intuitive presentation that include examples as well as comparison metrics. The organization is structured such that starting with an introduction to emerging technologies, up-to-date fundamentals of QCA are reported to engage the reader into the most recent advances of this field as reflected in the detailed treatment of sequential and combinational QCA circuits. The main emphasis is, however, on design and test to include digital QCA circuits and models for characterizing among the many attributes power consumption, defect diagnosis, modularity and fault tolerance. QCA can encompass multiple desirable features within different technological frameworks (based on metal as well as molecular implementations) and new computational paradigms (such as processing-by-wire and storage-by-motion).

The material covered in the chapters requires a basic understanding of physics, mathematics and electrical/electronic engineering, as commonly made available in an undergraduate degree program. This book can therefore be used as a reference as well as textbook for senior elective and graduate courses in nanotechnology, with an emphasis on emerging technologies. Advanced researchers will also find this book interesting as it provides a detailed treatment of QCA and issues involved in integrating basic device functionalities (combinational and sequential) into working circuits and systems. Novel research directions in QCA are also provided for the interested technical investigator. The authors of each chapter have an in-depth knowledge of QCA as reflected in their studies and work experience; this book is the result of the authors' research and development in QCA over more than five years as supported by federal agencies and industrial partners.

This book has been made possible by the collaboration of all authors; also, the authors would like to acknowledge enlightening discussions with Craig Lent (University of Notre Dame), Doug Tougaw (Valparaiso University), Konrad Walus (University of British Columbia), Cecilia Metra (University of Bologna), Salvatore Pontarelli (University of Rome Tor Vergata), Marya Libermann (University of Notre Dame), Niraj Jha (Princeton University), Hamid Hashempour, Sanjukta Bhanja (University of South Florida) and Jose Fortes (University of Florida). Their

insights and comments have been a tremendous encouragement for us to pursue the publication of this book.

Comments on this book can be sent to the editors by electronic mail: Jing Huang (hjing@ece.neu.edu) and Fabrizio Lombardi (lombardi@ece.neu.edu).

Jing Huang
Fabrizio Lombardi
Editors
Boston, Massachusetts
October 2007

Chapter 1

Introduction

J. Huang, M. Momenzadeh, and F. Lombardi

In the last few decades, the exponential scaling in feature size and increase in processing power have been successfully achieved by conventional lithography-based VLSI technology. However, this trend faces serious challenges due to fundamental physical limits of CMOS technology such as ultra-thin gate oxides, short channel effects, doping fluctuations and increasingly difficult and expensive lithography at nano-scale regimes. It is projected that the scaling process of known-today CMOS technology will end by the channel length of 7 nm by 2019 [1]. There has been extensive research in recent years at nano-scale to supersede conventional CMOS technology. It is anticipated that these technologies can achieve a density of 10^{12} devices/cm^2 and operate at THz frequencies [2].

Nanotechnology provides new possibilities for computing due to the unique properties that arise at such reduced feature sizes. Among these new devices, *Quantum-dot Cellular Automata* (QCA) [3] [4] relies on new physical phenomena (such as Coulombic interactions), and innovative techniques that radically depart from a CMOS-based model. QCA not only gives a solution at nano-scale, but it also offers a new method of computation and information transformation [5] [6]. Consider the processing features of CMOS systems: some circuits (i.e., logic gates) perform computation, while others (i.e., wires) are used for signal/data transfer and communication. In contrast, *computation and communication* occurs simultaneously in QCA [5]. QCA uses two basic logic gates, namely the INV and Majority Voter (MV). QCA is very promising because with this technology, computational paradigms which radically depart from traditional CMOS, can be implemented [7] [8] [9]. QCA design involves diverse and new paradigms such

as *memory-in-motion* and *processing-by-wire* [7] [10]. Memory-in-motion is an instance of the more general paradigm of processing-by-wire. Processing-by-wire (PBW) [10] is the QCA capability by which information manipulation can be accomplished, while transmission and communication of signals take place. PBW capabilities can be observed in the so-called inverter chain as well as in the arrangement of the cells in an MV. Besides the extra-high density feature, QCA can provide ultralow power dissipation and true power gain [11] [12] which are very promising due to the high density of this nano device. Recent development in QCA manufacturing involves molecular implementation. It is expected that molecular QCA will be manufacturing using DNA self-assembly and/or large scale cell deposition on insulated substrates [13].

1.1 CHALLENGES

The small size of QCA-based systems combined with their manufacturing methods (such as self-assembly) are substantially different from CMOS and make them more susceptible to defects and faults. In addition, defect in QCA manufacturing may well manifest themselves differently at logic level than CMOS. Defect characterization is therefore vital to design and test of QCA systems.

One of the fundamental issues in the testing community is the radical shift in computation and fabrication technology and its effect on the test flow. Do test generation and design-for-test become even intractable? Since the manufacturing process for nano devices is ill-defined, it is extremely difficult to address manufacturing testing problems. However, it would be inappropriate to ignore testing of these devices until the manufacturing state. QCA has the capability to provide defect tolerant operation and architectures that avoids massive logic redundancy or post-fabrication configuration. For QCA, placing individual cells on specific location on the substrate is difficult, and various types of cell misplacement defects may occur (such as cell misalignment, missing cell, or additional cell). These defects can have a substantial effect on the functionality of the device and hence the circuit. So proper testing of these devices for manufacturing defects plays a major role for quality of QCA-based circuits. Since the basic logic elements of a QCA-based design are different from conventional CMOS design, they need different testing schemes.

Moreover there are other manufacturing defects (such as faults in the clocking circuitry and the I/O mechanism) that may not occur during cell synthesis phase (in which the individual cells or molecules are manufactured) or deposition phase (in which the cells are placed in a specific location on the surface). Some of these faults

could separately be tested prior to QCA cell deposition, while others must be studied for modeling and characterization.

Because QCA system employs radically different computation paradigms, new design methodologies are needed to efficiently design large scale QCA systems. In QCA, the basic logic gate is the 3-input Majority Voter (MV), instead of the NAND, NOR gates in CMOS. Existing logic synthesis tool may not make use of MV efficiently. The quality of logic synthesis results when using existing tools need to be investigated. Additionally, there are no CAD tool available to directly translate QCA netlist into QCA layout. The lack of CAD support for QCA makes designing large logic systems extremely difficult, if possible at all. Design automation tool tailored to the unique features of QCA need to be developed.

The design and characterization of sequential circuits in QCA has not been fully addressed in the technical literature. While sequential elements can be implemented using QCA memory cells [7], such an approach would be prohibitive in terms of hardware (due to its extensive control circuitry) and very slow in performance. Moreover, sequentiality in QCA does not have the same requirements as in CMOS-based circuits. Latching is implicitly implemented in clocked QCA as sequential behavior is dependent on adiabatic switching and the layout of the QCA cells. The four-phase adiabatic clocking scheme for QCA introduces timing by dividing the QCA circuit into zones, and this unique feature imposes timing constrains on QCA sequential circuits. Methodology for designing sequential QCA circuits are required.

According to [13], QCA will likely be manufactured by self-assembly or large scale cell deposition on insulating substrates. These manufacturing technique are well suited for modular QCA design. However, these types of structured QCA design have not be investigated in detail.

1.2 PREVIOUS WORK

Previous work for defect tolerant QCA circuits has focused on individual cells and the majority voter (a basic logic element of QCA) [14] [15] [16]. A study of the fault tolerant properties of the majority voter under some manufacturing misalignments [17] [14] [15] show that the majority voter is more vulnerable to misalignment in the vertical direction than in the horizontal direction. A misalignment (at least equal to half a cell width in the vertical direction) causes the majority voter to malfunction. Based on this simulation-based study, a fault tolerant majority voter block has been proposed. In [16] Governale et al. have demonstrated that semi-conductor QCA is

sensitive to dot size and placement. Different dot size and misplacement should not be an issue in molecular QCA due to its structural nature.

In [18] [15], an $N \times N$ grid used as MV known as the Block Majority Voter, is analyzed. It has been shown that the Block Majority Voter is much more fault tolerant in terms of cell missing and cell misalignment defects compared to the regular MV. The possibility of designing fault tolerant QCA circuits has also been presented in [18]. Kogge et al. have shown in [19] that defects in a QCA wire severely affect its functional features; moreover it has been demonstrated that wider wires offer inherent defect tolerance [19] [15].

Combinational as well as sequential QCA design have been proposed, including circuits such as microprocessors [20], barrel shifter [21], SRAM [9] and FPGA [8]. In most published QCA sequential designs, sequential elements are implemented using memory cells, with the so called memory-in-motion technique [7]. In memory-in-motion, information is kept in a circulating loop controlled by the clock. An H-Memory architecture [7] which aims at high density and uniform access time has been proposed in [7]. In [9], a parallel memory architecture (similar to those encountered in CMOS-based RAM design) has been proposed for QCA. These memory architectures store information in a closed QCA wire loop, thus requiring a large number of clocking zones and complicating the underlying CMOS circuitry for providing the required clocking signals.

A modular methodology known as SQUARES has been proposed in [22]. In SQUARES, the basic building block is a 5×5 QCA cell grid. Logic gates, such as the MV and INV are directly embedded into the grid. The clocking assignment for SQUARES are quite complicated, as each grid is in its own clocking zone. No algorithm is given on how to efficiently assign clocking zones to SQUARES when designing a circuit.

Several QCA simulators, such as AQUINAS [23] and QCADesigner [24], have been developed. These tools perform an iterative quantum mechanical simulation (as a self consistent approximation) by factorizing the joint wave function over all QCA cells into a product of individual cell wave functions (using the Hartree-Fock approximation). These simulation tools can be used to investigate QCA design methodology.

1.3 CONTRIBUTIONS

In this book, the defect characterization of various QCA devices and the effect of these defects at logic-level have been extensively studied and investigated. Defect

injection is exploited to study the behavior of QCA-based circuits in the presence of defects and to measure the effectiveness of different test sets for detecting these defects. Unique testing properties of QCA technology have been identified and C-testability (where C stands for constant) of QCA designs based on majority voters is investigated. An efficient test generation approach has been proposed. The behavior of QCA devices in the presence of cell deposition defects is functionally modeled into erroneous logic behavior. Additionally, one of the goals of this work is to derive the likelihood of occurrence of functional faults in a QCA device using a layout driven method.

The defective behavior of QCA is well understood with respect to kink energy among off-center cells. However, no work has been reported on the behavior of defects with respect to variations due to scaling in the physical features of cells in QCA devices. Scaling plays an important role for QCA because it is related to its manufacturing process. For example, the relationship between a reduction in size and QCA cell placement is not yet fully understood for correct assembly. In this work, different fabrication schemes of various QCA devices at cell level are performed and the impact of various QCA cell sizes (scaling) in the presence of manufacturing defects is investigated. These different implementations are compared in terms of defect tolerance and testability.

QCA has been proposed as a possible physical technology to implement reversible computing [25]. A new mechanical model for QCA cells has been proposed that provides an intuitive and classical view of the energy and heat phenomena. This model can be used to analyze the energy consumption for a reversible computing system implemented using QCA technology. System-level defect tolerance schemes for reversible QCA circuits have been investigated in this book. The energy dissipation in QCA reversible circuits using the Maj-MUX fault tolerance technique is analyzed.

The traditional one-dimensional clocking scheme suffers from the disadvantage of long vertical lines in the placement of the cells, thus resulting in long delay, slow timing, the inability to operate at higher (room) temperature and sensitivity to thermal fluctuations. A two-dimensional QCA clocking scheme has been proposed in this book. The proposed clocking schemes are based on the equivalence between systolic processing and QCA zone switching. This technique results in a reduction in the longest line length in each clocking zone, permitting fast timing, efficient pipelining and kink-free behavior in switching.

Nanotechnology provides new possibilities for computing due to the unique properties that arise at such reduced feature sizes. Consider the processing features of CMOS systems: some circuits perform computation, while others are used for

signal/data transfer and communication. In FPGAs for example, computation is performed by the logic resources or PEs (processing elements), while communication is accomplished by the interconnect fabric (consisting of wires and switches in the channels separating the PEs). In QCA, *computation* and *communication* occur simultaneously [26] [7] [20]. This feature combined with the homogeneous cell arrangement capability of molecular QCA provides an opportunity for structured, modular QCA design. In this book, a modular approach based on elementary building blocks referred to as tiles, is proposed for QCA design. A tile is built using an $n \times n$ square grid of QCA cells. Different logic functions can be generated by using less than n^2 cells in a grid of dimension n. In particular, the 3×3 grid is shown to have unique properties which make it very attractive for synthesizing and designing larger circuits. Using different input and output cell arrangements, five tiles are analyzed as providing a high degree of flexibility in logic operation. The defect tolerance of QCA tiles has been analyzed by extensively studying the functional characterization of each tile in the presence of multiple undeposited cell defects. These features result in different combinational functions such as majority-like (with input inversion) and wire crossing capabilities. Examples of tile-based QCA design are presented in this book.

Sequential QCA design based on flip-flops is investigated in detail in this book. A novel RS-type flip-flop amenable to a QCA implementation has been proposed. This flip-flop extends a previous threshold-based configuration to QCA by taking into account the timing issues associated with the adiabatic switching of this technology. It is shown that an embedded QCA wire may lead to a D-type flip-flop behavior if it extends over multiple clocking zones.

In conventional logic design, synchronous operation is usually implemented in a sequential circuit. This circuit can be represented by a Mealy machine that consists of two parts: the flip-flops and the combinational logic. However for QCA, the four-phase clock signals control not only the flip-flops, but also the combinational gates. The entire QCA circuit is pipelined and latched by the clock signals. An important timing constraint in a QCA design is that for every logic gate all inputs must arrive at the same time, that is, all inputs must be in the same clocking zone (time matching). In synchronous sequential logic, all flip-flops compute at the same time. Therefore when designing this type of circuit in QCA, it is necessary to ensure that all paths from the outputs of the flip-flops (passing through the combinational logic) to the inputs of the flip-flops have the same delay (i.e., the number of clocking zones), thus *enforcing* the condition that signals arrive at the inputs of the flip-flops at the *same time* (strict matching). An algorithm for assigning appropriate clocking zones to a QCA sequential circuit is proposed. Examples

of QCA sequential circuits are provided. Additionally, defect characterization of sequential circuits is presented. Simulation results are provided for a logic-level characterization of the single additional and missing cell defects. It is shown that defects result in mostly unwanted inversion and stuck-at input values at logic level. Moreover, it is demonstrated that a device-level characterization of the defects and faults can be consistently extended to a circuit-level analysis.

Two novel memory architectures have been proposed in this book. The first one is a two-dimensional parallel memory architecture. The main advantage of this architecture is the sharing of the clocking zones between all memory cells in a column of the two-dimensional memory design. Therefore, the number of clocking zones for holding data is only dependent on the number of columns (word-size), that is, it is independent of the number of rows (memory-size). Also since clocking zones are shared, their dimensions are ideal to be clocked with underlying clocking circuitry. The second is a serial memory architecture. This architecture is based on utilizing building blocks (referred to as tiles) in the storage and input/output circuitry of the memory. A three-zone memory tile has been proposed by which information is moved across a concatenation of tiles by utilizing a two-level clocking mechanism. In the proposed memory, clocking zones are shared between memory cells and the length of the QCA line of a clocking zone is independent of the word size. QCA circuits for address decoding and input/output for simplification of the Read/Write operations have been discussed in detail.

The design of universal logic in QCA is also studied in this book. The universal gate is a logic gate that can implement any combinational function of its input variables. This type of gate is often used as a logic resource in array structures such as FPGAs. Logic design for the universal gate with three inputs is initially pursued using different synthesis techniques that are tailored to QCA. Next, as an alternative to universal gate, the QCA designs of various look-up-table (LUT) circuits are presented. These are either memory or multiplexer based circuits. Comparison between these arrangements is also pursued with respect to different figures of merit for universal design.

1.4 BOOK OUTLINE

Chapter 2 provides an overview of nanotechnology electronic devices. In Chapter 3 a review of QCA and a comparison of QCA with other nanotechnology devices are presented. Combinational QCA design is discussed in Chapter 4. Test generation and testability issue are discussed in Section 5.1. In Section 5.2, fault models and

defect characterization of QCA gates and interconnects, and their impacts on circuits are described and analyzed. In Chapter 6, a two-dimensional clocking scheme for high-performance QCA systems is proposed. Tile-based modular QCA design and the defect tolerance of QCA tiles are analyzed in Chapter 7. Chapter 8 presents flip-flop based QCA sequential design and defect analysis in sequential QCA circuits. Two new architectures for QCA, namely parallel and serial architectures, are presented in Chapter 9. The design of universal logic is investigated in Chapter 10. In Chapter 11, a QCA model is presented to analyze computation and energy dissipation, with focus on the possible application of reversible computing. Chapter 12 addresses the defect tolerance of reversible QCA circuits. Finally, conclusion and future work are addressed in Chapter 13.

References

[1] "International Technology Roadmap for Semiconductors," *Jointly Sponsored by European Semiconductor Industry Assc.,Japan Electronics and Information Technology Industry Assc., Korea Semiconductor Industry Assc., Taiwan Semiconductor Industry Assc., and Semiconductor Industry Assc.*, 2004.

[2] Lent, C. S. and B. Isaksen, "Clocked Molecular Quantum-Dot Cellular Automata," *IEEE Transactions on Electron Devices*,Vol. 50, No. 9, 2003, pp. 1890-1895.

[3] Lent, C. S., P. D. Tougaw and W. Porod, "Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules," *PhysComp '94: Proceedings of the Workshop on Physics and Computing*, IEEE Computer Society Press, 1994, pp. 5-13.

[4] Smith, C. G., "Computation Without Current," *Science*,Vol. 284, No. 5412, 1999, pp. 274.

[5] Amlani, I., et al., "Demonstration of a Six-Dot Quantum Cellular Automata System," *Applied Physics Letters*, Vol. 72, No.17, 1998, pp. 2179-2181.

[6] Orlov, A.O., et al., "Realization of a Functional Cell for Quantum-Dot Cellular Automata," *Science*,Vol. 277, No. 5328, 1997, pp. 928-930.

[7] Frost, S. E., et al., "Memory in Motion: A Study of Storage Structures in QCA," *1st Workshop on Non-Silicon Computation*, 2002.

[8] Niemier, M. T., A. F. Rodrigues and P. M. Kogge, "A Potentially Implementable FPGA for Quantum Dot Cellular Automata," *1st Workshop on Non-Silicon Computation*, Cambridge, MA, 2002.

[9] Walus, K., et al., "RAM Design Using Quantum-Dot Cellular Automata," *NanoTechnology Conference*,Vol. 2, 2003, pp. 160-163.

[10] Niemier, M. T. and P. M. Kogge, "Problems in Designing with QCAs: Layout=Timing," *International Journal of Circuit Theory and Applications*,Vol. 29, No. 1, 2001, pp. 49-62.

[11] Kummamuru, R. K., et al., "Power Gain in a Quantum-dot Cellular Automata Latch," *Applied Physics Letters*,Vol. 81, No.7, 2002, pp. 1332-1335.

[12] Timler, J. and C. S. Lent, "Power Gain and dissipation in Quantum-dot Cellular Automata," *Journal of Applied Physics*,Vol. 91, No. 2, 2002, pp. 823-831.

[13] Bernstein, G. H., et al., "Electron Beam Lithography and Liftoff of Molecules and DNA Rafts," *IEEE conference on Nanotechnology*, 2004, pp. 201-203.

[14] Armstrong, C. D., and W. M. Humphreys, "The Development of Design Tools for Fault Tolerant Quantum Dot Cellular Automata Based Logic," *2nd International Workshop on Quantum Dots for Quantum Computing and Classical Size Effect Circuits*, 2003.

[15] Fijany, A. and B.N. Toomarian, "New design for Quantum Dots Cellular Automata to Obtain Fault Tolerant Logic Gates," *Journal of Nanoparticle Research*, Vol. 3, No. 1, 2001, pp. 27-37.

[16] Governale, M.,et al., "Modeling and Manufacturing Assessment of Bistable Quantum-Dot Cellular Cells," *J. Appl. Phys.*, vol 85, No. 5, 1999, pp. 2962-2971.

[17] Armstrong, C.D., W.M. Humphreys and A. Fijany, "The Design of Fault Tolerant Quantum Dot Cellular Automata Based Logic," *11th NASA Symposium on VLSI Design*, 2003.

[18] Fijany, A., N. Toomarian, and K. Modarress, "Block qca fault-tolerant logic gates," Technical Report, Jet Propulsion Laboratory, California, 2003.

[19] Dysart, T. J., et al., "An Analysis of Missing Cell Defects in Quantum-Dot Cellular Automata," *IEEE International Workshop on Design and Test of Defect-Tolerant Nanoscale Architectures*, in conjunction with the VLSI Test Symposium, 2005.

[20] Niemier, M. T. and P. M. Kogge, "Logic-in-Wire: Using Quantum Dots to Implement a Microprocessor," *International Conference on Electronics, Circuits, and Systems (ICECS '99)*,Vol. 3, 1999, pp. 1211-1215.

[21] Dimitrov,V. S., G. A. Jullien and K. Walus, "Quantum-Dot Cellular Automata Carry-Look-Ahead Adder and Barrel Shifter," *IEEE Emerging Telecommunications Technologies Conference*, 2002.

[22] Berzon, D. and T. J. Fountain, "A Memory Design in QCAs Using the SQUARES Formalism," *Proceedings Ninth Great Lakes Symposium on VLSI*, 1999, pp. 166-169.

[23] Tougaw, P. D. and C. S. Lent, "Dynamic Behavior of Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 80, 1996, pp. 4722-4736.

[24] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also available online: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

[25] Lent, C. S., M. Liu and Y. Lu, "Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling," *Nanotechnology*,Vol. 17, No. 16, 2006, pp. 4240-4251.

[26] Amlani, I., et al., "Digital Logic Gate Using Quantum-Dot Cellular Automata," *Science*, Vol. 284, No. 5412, 1999, pp. 289-291.

Chapter 2

Nano Devices and Architectures Overview

J. Huang, M. Momenzadeh, and F. Lombardi

Conventional lithography-based VLSI technology (mostly utilizing CMOS) has been extremely successful in the last few decades, reducing feature size below 100 nm. As CMOS is fast approaching its fundamental physical limits (ultra thin gate oxides, short channel effects, etc.), new technologies at extremely small feature sizes (such as at nano scale) have been investigated to assess their viability for manufacturing future electronic/computing systems. New devices, such as carbon nanotubes, Si nanowires, single electron transistors, resonant tunneling diodes, single molecule devices, and spin transistors have been proposed [1]. It is projected that ultra-high density integration and ultra high speed operation can be achieved using these new devices.

Nanotechnology is a broad term that includes various areas of research such as electronics, chemistry, biology, physics, material science, and medicine. Here we focus on aspects of nanotechnology related to electronics. The National Science Foundation defines nanotechnology as having a feature size in the range of 1 to 100 nm to produce structures, devices, and systems with novel properties due to the reduced dimension. Devices that operate at nano scale, such as Field Effect Transistors (FETs), diodes, molecular and mechanical switches, have been recently built; moreover, non-volatile devices that hold their states in a few molecules, have been experimentally demonstrated [2] [1]. Different techniques have been shown to be effective in the assembly of nanometer wide wires into large arrays [3] [4]. At this reduced size, systems require completely new approaches to manufacturing and fabrication with immediate implications and significant impact on circuit design and architectures. Currently, semiconductor technology uses a “top-down” approach

that lithographically imposes a pattern. Unnecessary bulk material is then etched away to generate the desired structure. An alternative process to avoid the sophisticated and expensive nano-scale lithography is to use a so-called self-assembly, in which the nanostructures can be spontaneously built, i.e., self-assembled from the "bottom" on a molecule to molecule basis.

These chemical self-assembly processes are expected to considerably lower manufacturing cost. However, these "bottom-up" techniques will likely result in much higher defect rates then conventional top-down lithography [5]. Thus, it is probable that in the future, these devices will be less defect tolerant than present day devices. It is suggested in [5] that the very nature of chemical self-assembly based fabrication will result in defect densities of as much as 10%. Additionally, these new devices are expected to be more sensitive to the external environment (such as electromagnetic interference, thermal fluctuations and radiation related effects) [6], thus resulting in a higher rate of soft errors. So, it is widely expected that a large percentage of manufactured devices will be defective. If progress must be made in nanoelectronics, fault-tolerant architecture will certainly be required to produce systems that are resilient to manufacturing defects and transient errors. These circuits should have some Build-In-Self-Test (BIST) structures that allow self test/diagnosis, and use redundancy to bypass faults. Fault tolerance strategies for nanotechnology have been investigated in [7] [8] [9]. In [7], the authors proposed techniques to bypass defective resources during logic mapping. These techniques are applicable to nanoscale crossbar structures by taking advantage of the inherent redundancy. The Recursive NanoBox Processor Grid has been described and evaluated in [9] as a defect tolerance scheme for parallel computing systems. Reference [8] deals with dynamic fault tolerance of crossbar-based nanoscale memory. In addition, architectures based on programmable PLA-like arrays have already been proposed [10] [11], by which reconfigurability is used to achieve defect-tolerance.

2.1 NANOELECTRONIC DEVICES

2.1.1 Carbon Nanotube-based Devices

Carbon Nanotubes (CNTs) [12] can be visualized as sheets of graphite rolled into seamless cylinders of nanometer diameter and micron scale length (as shown in Figure 2.1). As molecular-based devices, CNTs are extremely strong, flexible and transfer heat very efficiently [13]. Depending on their chirality (i.e., the lattice structure), CNTs can be metallic or semiconducting. The tubes can be made into

single-walled nanotubes (SWNT) or multi-Fwalled nanotubes (MWNT) as multiple SWNTs wrapped over one another [14].

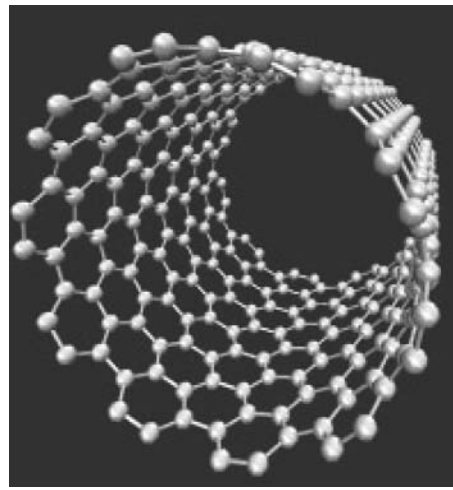


Figure 2.1 A Single Walled Carbon Nanotube (from [15]. ©2004 IEEE. Reprint with permission)

It has been shown that CNTs can be used as molecular wires and scanning probe microscopy and lithography [16] [17] [18], diodes [19], field-effect transistors (FETs), SETs, programmable switches [20], memory [21] or energy storage for batteries and fuel cells [22]. However currently there is no known synthesis procedure to produce a pure batch of just one type (metallic or semiconducting) [2] of CNTs. This makes specific device fabrication a likely random process and it poses severe limitations on integrating large systems.

An enhancement-mode p-type FET built with a single CNT has been demonstrated in [20]. This gate consists of an Al wire (as gate) over a negative Al_2O_3 layer of only a few nanometers in thickness, that lies beneath a single CNT (as conducing channel). This CNT FET has been used to build various logic circuits such as an inverter, NOR gate and SRAM cell [20]. However, the process by which semiconducting nanotubes are placed on specific locations on the wafer, still remains very difficult to solve [20]. Without special processing, CNT FETs exhibit p-type characteristics. It has been shown in [23] that n-type CNT FETs can be manufactured by doping, or annealing p-type CNT FETs in vacuum. An inverter made of both p-type and n-type CNT FETs has been demonstrated in [20], and shown in Figure 2.2. Metallic CNTs have been shown to be ballistic. In ballistic transport, charge carriers

driven by electric fields move in a conducting or semiconducting material without scattering. [24] has shown that by using Pd contacts, a ballistic CNT FET can be built such that the "ON" state of semiconducting CNTs can behave as ohmically contact ballistic metallic CNTs.

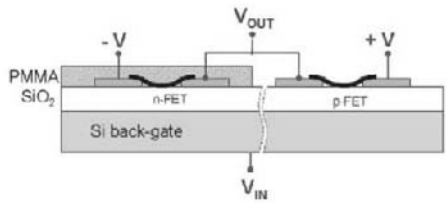


Figure 2.2 CNT Inverter (From [20]. ©2001 Science. Reprint with permission)

In [19], characteristics of junctions consisting of two CNTs was analyzed. These junctions are formed by laying one CNT across the other. Individual CNTs are identified as metallic (M) or semiconducting (S); MM,SS,MS junctions have been proposed [19]. It has been shown that MM and SS junctions have high conductance, while a MS junction acts as a rectifying Schottky barrier diode.

In [21] a suspended, crossed nanotube geometry has been utilized for bistable programmable switches; this structure will be discussed in greater detail in Section 2.2.

2.1.2 Nanowires

A big limitation of CNT is the inability in manufacturing to control whether the CNT is metallic or semiconducting. This poses a significant difficulty for large scale device fabrication. Single crystal silicon Nanowires (NWs) have been fabricated, with diameter ranging from 6 to 20 nm and length from 1 to 10 microns [2]. Unlike CNTs, the electronic properties of NWs can be precisely controlled during synthesis [25]. Metallic as well as semiconductor NWs have been demonstrated. These devices can be used to build wires, diodes and FETs [2] [26] [25] [3]. Unlike from CNTs, NWs can be controlled very accurately during synthesis and methods exist for parallel assembly at manufacturing. Si NWs can be doped using phosphorus and boron to have either p-type or n-type devices [26].

It has been demonstrated [25] that a pn junction can be formed by crossing a p-type silicon NW and n-type gallium nitride (GaN) NW; this junction exhibits current

rectification characteristics with a typical turn-on voltage of $1.0V$. Experimental results have shown that this NW cross junction has a yield of 95%. A bipolar transistor that consists of n+ and n-type NWs crossing a common p-type wire, has been constructed in [26]; this transistor has a common base gain of 0.94 and a common emitter gain of 16. Furthermore, the n-GaN/p-Si cross NW junction with high turn-on voltage can be used as an FET [25] (shown in Figure 2.3). The high turn-on voltage is obtained by growing an oxide layer to prevent direct electrical contact of crossed conductors, thus obtaining junctions that exhibit FET behavior [25]. Logic gates can be fabricated using these cross junction FETs (shown in Figure 2.4). In [25], an AND gate has been fabricated from one p-Si and three n-GaN multiple junctions. Diode resistor logic is used, as shown in Figure 2.4(a). Three n-GaN NWs (horizontal) and one p-Si NW (vertical) is used. Two of the GaN NWs are used as inputs, while the third GaN NW (with constant voltage) acts as a resistor by depleting a portion of the p-Si NW. The NW FET junctions are used to build a NOR gate [25], as shown in Figure 2.4(b). The gate has a p-Si NW (as conducting channel) and n-GaN NWs (as gates). A voltage gain of 5 has been reported for this gate [25].

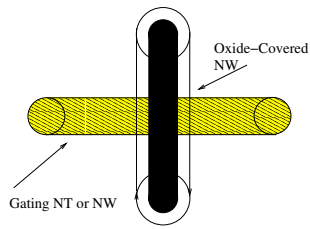


Figure 2.3 NW FETs

NW-based crossbar structures have been fabricated in [4] [3] [27] [28]. This will be discussed in detail in Section 2.2.

2.1.3 Molecular Electronic Devices

Besides CNT, work has been reported on using single molecules to build electronic devices. Molecular electronic devices (such as tunneling junctions, rectifiers, single-molecule transistors and programmable molecular switches) have been analyzed in [29] [1]. Molecular electronic devices are attractive, because a molecule has a size range from 1 to 100 nm, a scale that permits functional nanostructure with advantages in cost and efficiency [30]. Also, inter-molecular interactions may be

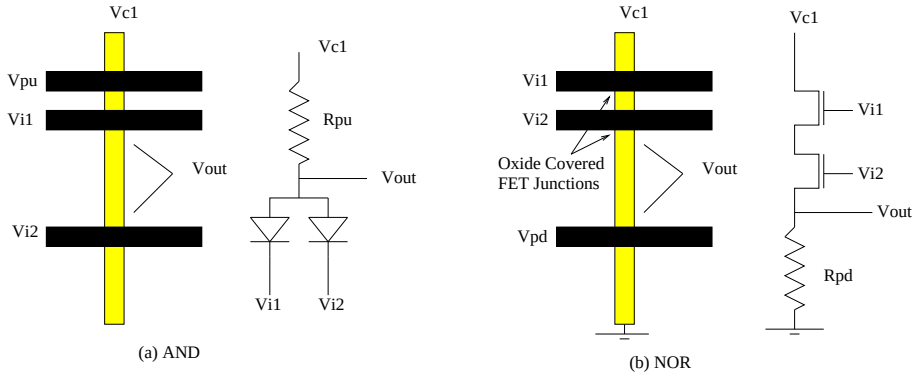


Figure 2.4 NW Gates

used to form structures by self-assembly, thus making them cost-effective. However, molecules have disadvantages, such as instability at high temperatures. Furthermore, the characteristics and performance of molecules need to be understood not only in the solution phase, but more importantly in the solid-state phase.

Tunneling junctions are built with linear alkanes sandwiched between metal electrodes [29]. A molecule composed of an electron donor, a bridge, and an acceptor (extended between two electrodes) has been shown to exhibit rectifying behavior [29] [30]. A single molecular transistor is depicted in Figure 2.5(a). The molecule acts as a conducting channel and is bridged across a 1 to 4 nm wide electrode gap [29]. In single-molecule transistors, a unique type of quantum mechanical resonance (namely the Kondo resonance) has been observed [29]. Molecular transistors can not qualitatively provide new performance characteristics compared to conventional FETs [1], but they may offer better performance through improved material parameters and manufacturing schemes. Programmable switches have also been built with molecules [31] [29] [32]. This switch can hold its own state and can also be programmed by signal wires for crossing [31]. Bistable molecules (such as catenanes and rotaxanes), can be used as switches. The two states of the molecule correspond to the "ON" and "OFF" states of the switch. Switching from one state to another is accomplished by applying an appropriate voltage. Figure 2.5(b) shows a molecular switch built with rotaxane, and its structural formula in the "ON" state. In [31], imprint lithography is used for a molecular switch that consists of a monolayer of bistable rotaxanes sandwiched between two 40 nm electrodes. For 75% of the devices tested, reversible switching properties have been verified.

The resistance of the "ON" state is $R_{on} < 10^5\Omega$, while the resistance of the "OFF" state is $R_{off} > 10^8\Omega$ [31]. The switches can be moved between the two states by applying $\pm 0.5V$ to $\pm 3V$ as programming voltage. Experimental results have shown that the ratio between R_{on} and R_{off} typically decays below 2 and gradually approaches 1 after a few to several hundred cycles of programming [31].

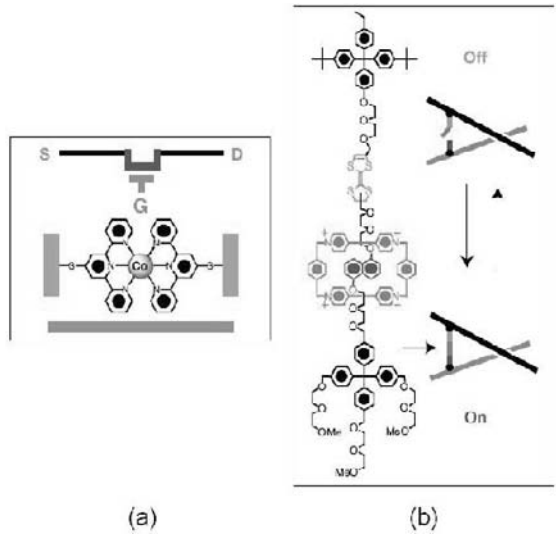


Figure 2.5 (a) Molecular Transistor (b) Programmable Molecular Switch (From [29]. ©2004 Science. Reprint with permission)

2.1.4 Single-Electron Devices

In single-electron devices, the motion of each electron is controlled individually via tunnel barriers. To exhibit quantum behavior, an island associated with a tunnel barrier needs to be very small in size, so that a single electron that is added to the island, can cause a significant voltage increase [33]. Electron tunneling through a particular barrier has been formulated by the so-called *orthodox theory* presented by Averin and Likharev in [34].

Single-electron tunneling devices consist of a single-electron box, a single-electron transistor (SET), a single-electron trap and a single-electron turnstile and pump.

2.1.4.1 Single-Electron Box

A single-electron box is based on a small island separated from a larger electrode *electron source* by a tunnel barrier (as shown in Figure 2.6). An external electric field can be applied to the island using another electrode (or gate) separated from the island by a thicker insulator, that does not allow noticeable tunneling [35]. The field controls the conditions of electron tunneling by changing the electrochemical potential of the island.

The disadvantages of the single-electron box are the lack of internal memory (the number of electrons in the box is a unique function of the applied voltage) and the inability of carrying DC current (an ultrasensitive electrometer is necessary to measure its charge state) [36].

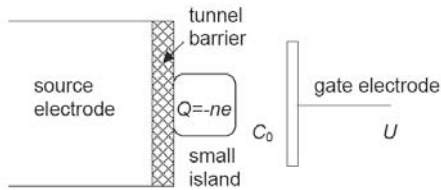
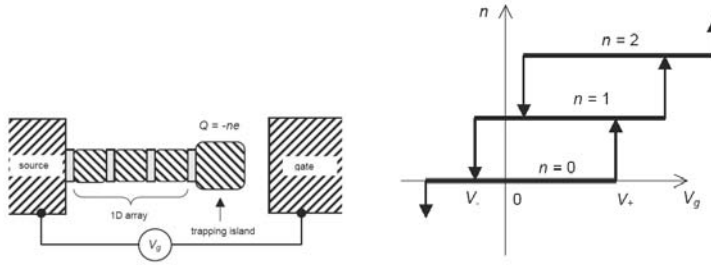


Figure 2.6 Single-Electron Box Schematic Diagram (From [35]. ©1999 IEEE. Reprint with permission)

2.1.4.2 Single-Electron Trap

The previously described drawback of a single-electron box is corrected in a single-electron trap. A single-electron trap [37] [38] can be obtained by replacing a single tunnel junction (as a generalization of the single-electron box) with a one-dimensional array of islands separated by tunnel barriers [35], as shown in Figure 2.7(a). This structure provides an internal memory configuration; for certain ranges of V_g (between V_+ and V_-) the system may be in one or more charge states of the trapping island (as shown in Figure 2.7(b)) [35]. Electron retention of more than 12 hours at very low temperature has been experimentally demonstrated in [39] and [40].

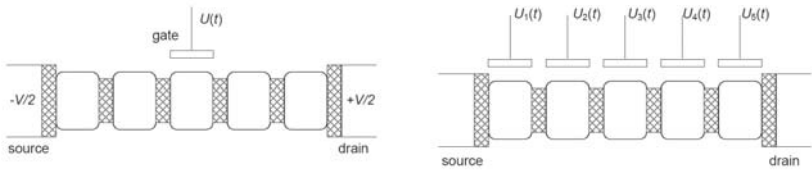


(a) (b)

Figure 2.7 (a) Schematic Diagram (b) Static Characteristics at $T \rightarrow 0$ of a Single-Electron Trap (From [41]. ©1999 Nano Letter. Reprint with permission)

2.1.4.3 Single-Electron Turnstile and Pump

The single-electron turnstile methodology is a combination of a single-electron box and a single-electron trap [35], as shown in Figure 2.8(a) [42]. When $V = 0$ the device acts as a single-electron trap; an electron may be pulled into the island, resulting in an increase of the voltage U; then, it may be pushed out by decreasing U. If $V \neq 0$, an electron is received at the source (when U increases) and delivered to the drain (when U decreases) [35].



(a) (b)

Figure 2.8 Schematic Diagram of Single-Electron (a) Turnstile (b) Pump (From [35]. ©1999 IEEE. Reprint with permission)

In a single-electron pump [43] (as shown in Figure 2.8(b)) the signals $U_i(t)$ that are applied to each electrode are phase-shifted to form a potential wave gliding along the island array, leading an electron from source to drain.

2.1.4.4 Single-Electron Transistor (SET)

The latter drawback of a single-electron box can be corrected by splitting the tunnel junction and applying a DC voltage between the two electrodes, as shown in Figure 2.9.

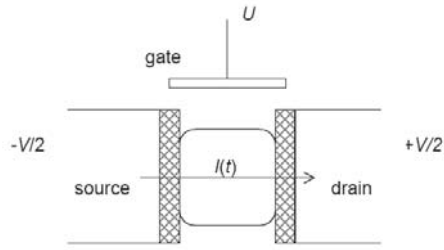


Figure 2.9 Schematic Diagram of Single-Electron Transistor (From [35]. ©1999 IEEE. Reprint with permission)

The significant structural feature of an SET is a small island (dot) made of a semiconductor, or metal in which electrons can be confined. An SET consists of three terminals and operates on Coulomb blockage [44] [45]. A gate controls the number of electrons on the dot (Figure 2.10). The energy that must be placed or removed from the dot depends on the size of the dot (1 to 3 nm at room temperature [1]) and the number of electrons that are already in it. Among single-electron tunneling devices, SETs are the most popular devices due to their similarities to MOSFETs.

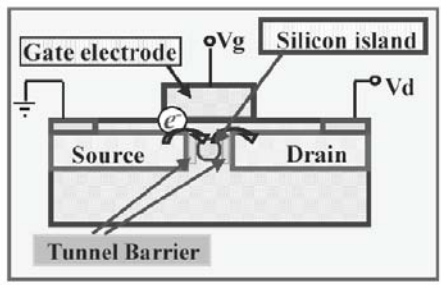


Figure 2.10 Single-Electron Transistor Structure (From [46]. ©2000 IEEE. Reprint with permission)

2.1.5 Resonant Tunneling Diodes

The resonant tunneling diode (RTD) [54] [55] is an extremely fast device with measured slew rates as high as 300 mV/ps [56]. The RTD is made of a sandwich of two very thin layers of high-band-gap material (acting as potential energy barriers – source and drain) surrounding a thin layer of lower band-gap material [57]. This device is characterized by a region of negative differential resistance (NDR) in the I-V curve, as shown in Figure 2.11(a). The local maximum (minimum) in the current is called the peak current or I_P (valley current or I_V) , occurring at the peak voltage V_P (the valley voltage V_V). The current falls off above the peak voltage reaching a minimum, before rising again due to scattering and bias-induced lowering of the barriers.

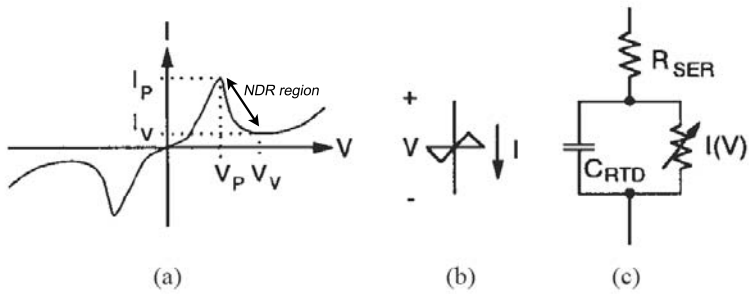


Figure 2.11 RTD (a) I-V Curve (b) Schematic and (c) Equivalent Circuit (From [57]. ©1999 IEEE. Reprint with permission)

The NDR of an RTD not only provides amplification, but it also results in another important feature, namely the multi-peak I-V characteristics that are obtained when several RTDs are combined in series. The nonlinear characteristic of an RTD provides the opportunity for its use in a wide class of circuit applications, such as multivalued logic, nanopiplined high-speed circuits and circuits with low power-delay products [58]. A molecular scale latch that is based on RTDs is proposed in [59]. Molecular devices (with a so-called peak-to-valley ratio as figure of merit) have been reported for room temperature operation [60].

Integration of a transistor with a pair of RTDs initiates delay issues as operational speed of an integrated device can be an order of magnitude slower than those of RTDs due to capacitive charging and discharging of a transistor gate [61].

Another issue is limitation on scaling due to low dynamic range of 10 for RTDs compared to required factor of 10^5 enjoyed by CMOS designers [61].

2.1.6 Spin Transistors

Conventional transistors as well as the previously presented nano devices (such as CNT, NW, SET) are based on the charge that electrons carry. Since an electron has not only charge, but also spin, a spin FET has been proposed in which information is now carried by the spin of the electrons [1] [62]. Electron spin is a fundamental unit of magnetic moment, which provides the basis for magnetic memories. A spin FET consists of a ferromagnetic source and a drain. Spin-polarized electrons are injected into a quasi one-dimensional semiconductor channel from the source [62]. The electrons propagate through the channel to the drain. The probability of a electron exiting the drain is dependent on the relative orientation of the spin of the electron and the drain's fixed magnetization [62]. By applying a gate voltage, it is possible to rotate the electron spin, thus controlling the drain current. A possible spin transistor structure that has been proposed in [62], is depicted in Figure 2.12.

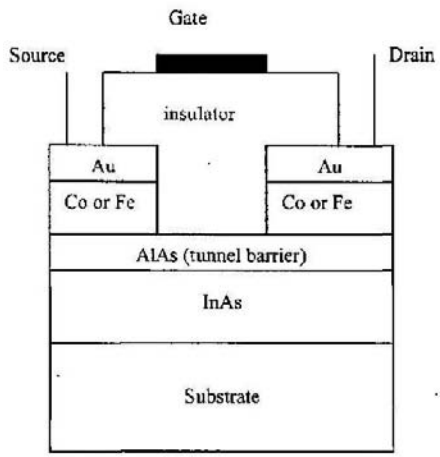


Figure 2.12 Spin FET Structure (From [62]. ©2004 IEEE. Reprint with permission)

Spin FETs promise a faster switching speed and a lower energy dissipation than conventional MOS FETs. However, such a device has not yet been built due to several major challenges that spin FETs still face. The most recognized one is how

to inject spin-polarized electrons from the ferromagnetic into semiconductors due to the resistance mismatch between these two materials [62] [1]. Another obstacle is represented by Ramsauer resonance [62]. If the barriers between the channel and the contacts have an abrupt potential change, an electron in the channel will be "reflected" between the contacts multiple times before exiting the channel. The energy level in the channels will be quantized. In this case, when the gate voltage sweeps the Fermi level through a quantized energy level, the conductance will exhibit a resonance peak. These resonances are referred to as Ramsauer resonances [62]. Furthermore, the magnetic field in the channel introduces a new type of spin relaxation mechanism [62] such that non-magnetic scatterers can flip spin. Research is being pursued on techniques for combining ferromagnetic metals and semiconductors [1].

2.2 NANO-SCALE CROSSBARS

CNTs and NWs can be made into a nano-scale crossbar structure that has been suggested as a promising candidate as a basic building block of nanoelectronics circuits [63] [10] [3] [21] [28] [64]. A nano-scale crossbar consists of two sets of parallel nano-scale wires, perpendicularly crossing each other. The wire crossings form junctions that can be a programmable switch, a diode, or an FET [63] [10] [21]. Nano-scale crossbars are attractive for several reasons. It is expected that large scale nanoelectronics circuits will heavily rely on bottom-up approaches for manufacturing. In this methodology initially, individual devices and wires are manufactured, subsequently individual devices are assembled into components, and components into larger units. These units are then connected into a complete system. Many different techniques for assembling and aligning nano-scale components exist [2]. The common feature of these self-assembly techniques is that they can only form simple, regular structures, such as crossbars. Further more, as explained in Section 2.1, various devices, such as switches, diodes and FETs can be formed at the cross junctions of NWs and CNTs. It has been shown that crossbars can be used as memory and programmable logic arrays as well as interconnect fabrics [63] [10] [21]. For example, crossbars with programmable crosspoint diodes can be used as programmable OR arrays, using resistor-diode logic [10]. Crossbar structures have been shown to be defect tolerant [10] [11]. The cross junction device elements are addressable within a large array; high level architectures based on the crossbar structure, have been proposed in [10] [11].

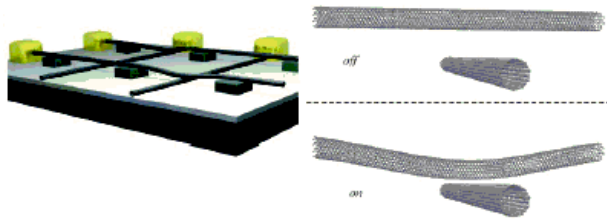


Figure 2.13 Suspended CNT Switch Crossbar (From [21]. ©2000 Science. Reprint with permission)

A programmable switch that is based on a suspended, crossed SWNT is proposed in [21]. This leads to a bistable switch, electrostatically switching between ON/OFF states, as shown in Figure 2.13. The CNT-CNT junction is bistable with an energy barrier between the two states. In the first state, the tubes are "far" apart and the mechanical forces keep the upper CNT away from the lower CNT. This is referred to as the "off" state. At this distance, the tunneling current is very small, thus resulting in a resistance in the order of $G\Omega$. In the second state, the tubes come into contact and are held together by the van der Waals force. In this state, there is little resistance between the tubes. By applying a voltage to the tubes, it is possible to change them to the same or opposite polarity. Attraction/repulsion in electrical charges is utilized to cross the energy gap and thus programming this device to the on/off state. By using semiconductors CNTs or NWs for the lower molecular wire, it is then possible to have a rectifying diode at the crossing point in the "on" state [21].

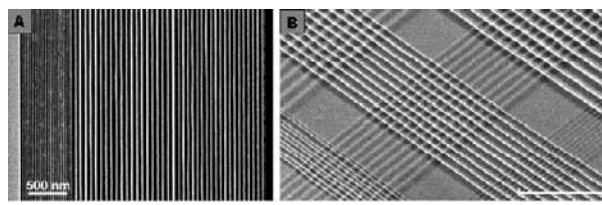


Figure 2.14 Parallel NWs and NW Crossbars Manufactured (From [3]. ©2003 Science. Reprint with permission)

Fabrication of crossbars by NWs has been reported in [27] [3] [4]. A solution-based method has been demonstrated in [4]. Silicon NWs in solution are aligned

and then transferred to the surface of the substrate to form a parallel NW array with controlled spacing [4]. The process is then repeated to transfer a second layer of aligned NWs perpendicular to the first layer. Photolithography can then be used to define a pattern on the substrate; all NWs outside the pattern are then removed by gentle sonication. The resulting circuit consists of $10\mu m \times 10\mu m$ NW crossbar arrays, with $25\mu m$ array pitch. Within each array, there are $40nm$ NWs, with $500nm$ NW spacing. A superlattice NW pattern transfer (SNAP) technique has been proposed in [27] [3] to build NW crossbars. Figure 2.14(a) show the 40 platinum (Pt) NWs with $10nm$ diameter, $60nm$ spacing and 20 Pt (platinum) NWs with $10nm$ diameter and $30nm$ spacing. Figure 2.14(b) shows a Pt NW crossbar manufactured in [3]; the spacing between NWs ranges from $20nm$ to $80nm$.

As explained in Section 2.1, bistable molecules can be used to make programmable switches. A nano-scale 8×8 crossbar that consists of a molecular monolayer of bistable rotaxanes sandwiched between metal wires, has been manufactured in [63]; it occupies a $1\mu m^2$ area. Each crosspoint can be used as an active, non-volatile memory cell. 85% of the manufactured switches have shown switching behavior. A voltage of $3.5V$ to $7V$ writes a "1" to the memory cell, while a voltage of $-3.5V$ to $-7V$ writes a "0" in the memory cell. Furthermore, the 8×8 crossbar has been programmed into a 4×4 memory array, a 4×4 demultiplexer, and a 4×4 multiplexer decoder. External circuits are needed to connect the decoders with diodes and capacitors.

Nano-scale circuit design with crossbars has been investigated in [64]; both resistor-based and diode-based junctions have been considered by mapping logic resources to a crossbar via a programmed decoder. An interesting feature of these devices is that they have the ability to store state values and implement programmable switching at wire crossings. For example, the programmable switch in an FPGA consists of a pass transistor and a SRAM cell for the configuration information, thus requiring a substantial amount of chip area. With molecular wires, a programmable switch occupies the space of only a primitive wire crossing, thus permitting a fully populated switch at a small impact on area.

2.3 ARCHITECTURES

Several higher level architectures based on nano-scale devices and structures (as introduced in the previous sections) are discussed next.

2.3.1 SET Architecture

An SET-based architecture has been proposed for SRAM in [47] and shown in Figure 2.15. This architecture is composed of two crossconnected SETs (TI and T2) and exhibits Negative Differential Resistance (NDR) characteristics. No capacitor is used for information storage, hence making this architecture a candidate for highly dense SRAM structures [47].

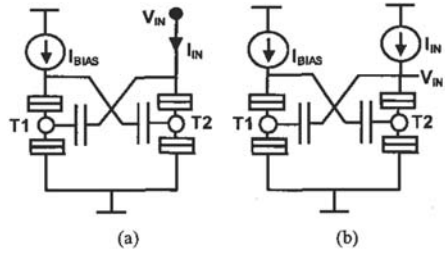


Figure 2.15 Schematic Diagram of SRAM Architectures for (a) Negative Differential Resistance (NDR) (b) Hysteresis Effect (From [47]. ©2004 IEEE. Reprint with permission)

T1 is biased by a constant current source. If T2 is biased by a voltage source, the feedback loop created by the current source and T1 decreases the gate-to-source voltage of T2 (and consequently decreasing the drain current I_{IN}) when the input voltage (V_{IN}) is increased. Therefore, the input conductance ($g_{in} = dI_{in}/dV_{in}$) will have Negative Differential Resistance (NDR) characteristics. If T2 is biased by a current source (as shown in Figure 2.15(b)), hysteresis characteristics can be observed by utilizing the NDR property [47].

2.3.2 RTD Architecture

RTDs are being used to build SRAM cells. Also in [59], a molecular scale latch that provides signal restoration, is proposed for implementation by considering interactions among a pair of molecular RTDs [57] [65]. Figure 2.16(a) depicts a bistable latch when voltage is biased within a suitable range (beginning at about $2 \times V_P$), as shown in 2.16(b). The pair is monostable if the system is biased below this range. The state of a bistable pair is given by the voltage of the data node. The grounded RTD (drive RTD) is biased through the load RTD. The data node voltage is represented as high (“1” state) and low (“0” state). This latch is constructed using a nanowire that includes the RTD molecules within the wire.

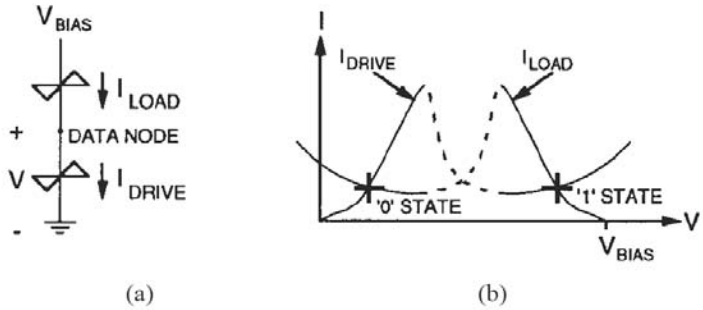


Figure 2.16 An RTD Latch (a) Schematic Diagram (b) Load-line Diagram (From [57]. ©1999 IEEE. Reprint with permission.

2.3.3 NanoFabrics Architecture

A chemically assembled electronic nanotechnology (CAEN) based architecture has been proposed in [11]. This architecture is similar to an FPGA. CAEN is a form of electronic nanotechnology that uses self-alignment to construct electronic circuits out of nano-scale device. Since CAEN is highly unlikely to produce complex aperiodic structures, the architecture introduced in [11] is based on fabricating dense simple regular structures, which are called nanoBlocks, and can be programmed to generate the desired functionality. The array of connected nanoBlocks is referred to as *nanoFabrics*.

The structure of a nanoblock is shown in Figure 2.17, each nanoblock is based on a molecular logic array (MLA). The MLA is constructed with two layers of parallel NWs crossing each other at right angle. At each intersection of the NWs is a programmable switch. When the switch is programmed on, it acts as a diode. Diode-resistor logic is used to implement the logic functions. To create a complete design, signals and their complements are brought into each circuit to generate both the desired functions and their complements. For example, an AND gate implemented in the MLA is illustrated in Figure 2.18. However, the switches are passive devices, therefore some sort of signal restoration is required in the block. This is achieved by using a molecular latch composed of a wire with two inline NDR molecules, as described in the previous section.

The nanoblocks are then organized into clusters. The outputs of one nanoblock are connected to the inputs of another by crossing two groups of orthogonal wires,

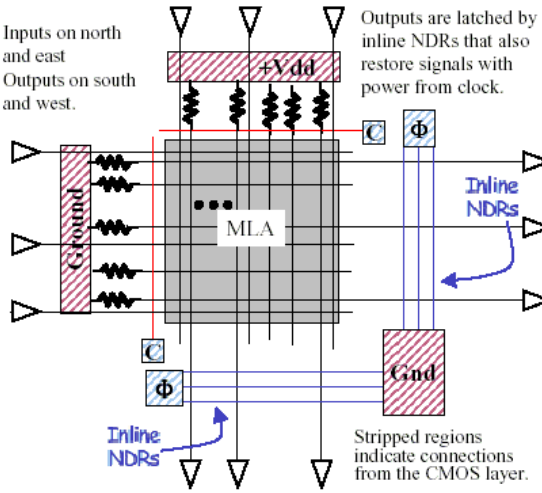


Figure 2.17 NanoBlock (From [11]. ©2001 IEEE. Reprint with permission)

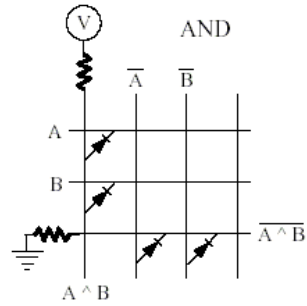


Figure 2.18 An AND Gate Implemented in MLA (From [11]. ©2001 IEEE. Reprint with permission)

so that no precise end-to-end alignment is needed. The connections of the nanoblock are shown in Figure 2.19. The area in which the input and output of the blocks overlap, is referred to as a switch-block. Long lines run between clusters to provide low-latency communication over long distance. The whole structure is similar to an island-style FPGA. The complete structure is shown in Figure 2.20.

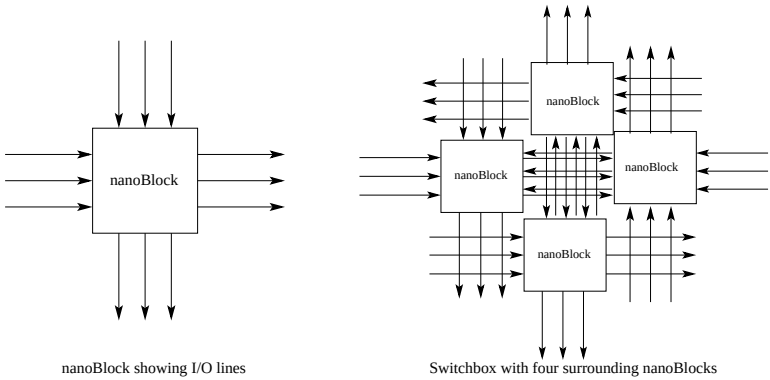


Figure 2.19 Connecting NanoBlocks (From [11]. ©2001 IEEE. Reprint with permission)

The nanoFabric is defect tolerant because it's regular, fine-grained and configurable [11]. [5] has proposed a scalable testing methodology for finding defects in reconfigurable devices. This paper addressed the problem of finding the defects in a nanoFabric. After the defects are located, the CAEN-built reconfigurable fabric can be reconfigured to achieve fault tolerance. The test/diagnosis scheme involves configuring the nanoFabric into a number of tilings (configurations). In each particular tiling, the components are configured into test circuits. If a test circuit passed the test, all components in that test circuit are marked to be fault-free. The above procedure is repeated for many tilings, so that each component is part of many different test circuits. At the end, all the components marked fault-free can be used in mapping desired function.

2.3.4 NanoPLA

An array-based architecture that is similar to a PLA has been proposed in [10]. This structure is based on self-assembled crossed arrays of NWs with non-volatile diode switches at the intersections. Signal restoration is accomplished using NW FET devices as buffers. A stochastic approach to addressing individual NWs without

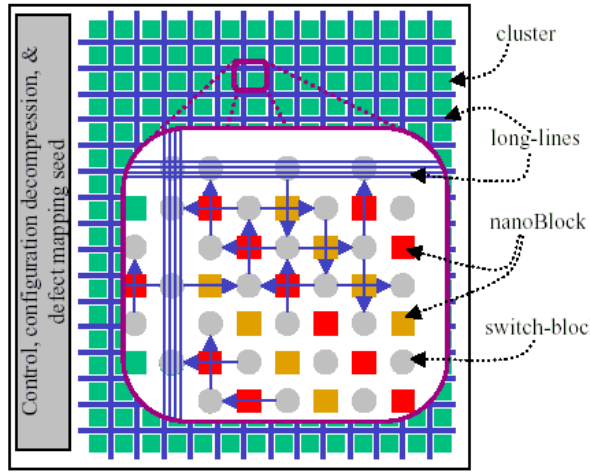


Figure 2.20 Layout of the NanoFabrics (From [11]. ©2001 IEEE. Reprint with permission)

using lithography-scale processing at a nano-scale dimension, has been described in [66]. A programmable interconnecting architecture to connect the nanoPLA blocks has been discussed in [67].

As mentioned in previous sections, NWs can be built into crossbars with programmable diodes at the cross junctions. They can be used as programmable wired-OR arrays as shown in Figure 2.21. Diode-resistor logic is used. However, only passive devices are used, so this array can not provide gain; proper signal restoring logic is needed. Also the OR function is not universal; to implement arbitrary logic inversion is also needed. Using doped NWs, FETs can be built. For example, a NOR gate is shown in Figure 2.4. FET buffers or inverters can be placed between diode stages to provide both signal restoration and inversion when desired.

The basic tile of the architecture proposed in [10] with programmable OR arrays is shown in Figure 2.22. Precharge/evaluation logic is used in this architecture. Following each programmable logic OR array, fixed inversion/buffer arrays built with NW FET devices are used to restore signal levels as well as providing logic inversion. Since doped NW behaves as FETs, conduction along the NW can be controlled by an applied electric field [66]. Modulation-doped NWs can therefore be coded and used to address individual NWs. [66] has proposed building a stochastic address decoder, such that NW addressability can be obtained without relying on nano-scale lithography. The basic tile can then be interconnected to build the entire

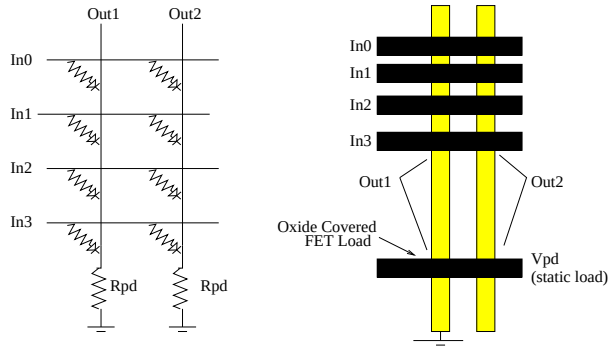


Figure 2.21 Programmable Diode OR Array

programmable logic array. Programming voltages must be higher than the operating voltages for the FET or diode logic. Therefore different voltages are placed on the decoder's supply voltage.

Recently, a three-dimensional NanoPLA architecture has been proposed in [68]. The basic structure of this architecture is similar to the original NanoPLA shown in Figure 2.22. The difference is that 3D NanoPLA is realized from layers of semiconducting NWs stacked on top of each other. A possible manufacturing scheme is discussed in [68] and it is shown that by using a compact three-dimensional layout, a reduction in delay of 18% is achieved.

Defect tolerance issues have been also discussed in [10]. Further more, manufacturing yield of the proposed array-based architecture has been analyzed. This study is based on the NW defect density by estimating the yield of the stochastic decoder and the stochastic buffering.

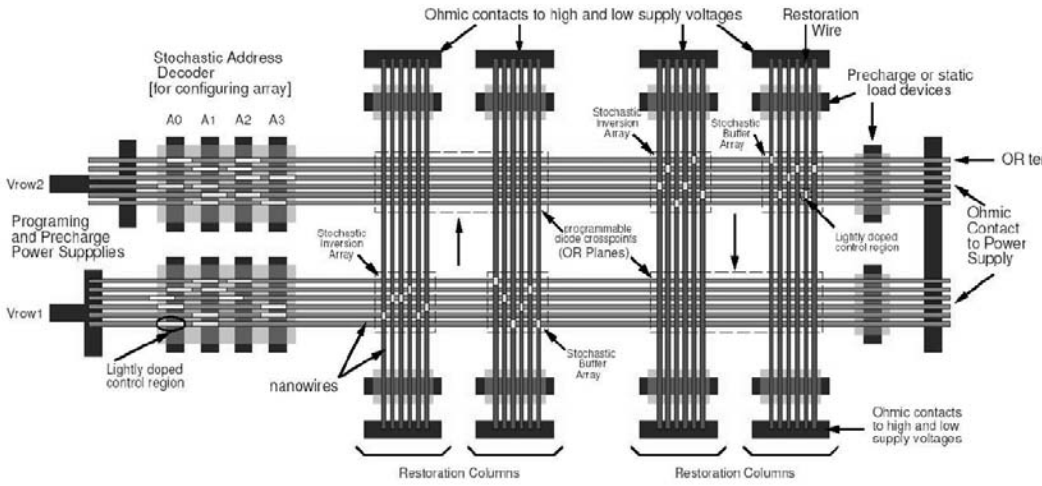


Figure 2.22 PLA-like Nanowire Programmable Logic Array Tile (From [10]. ©2004 ACM. Reprint with permission)

References

[1] Lundstrom, M., "Is Nanoelectronics the Future of Microelectronics?," *International Symposium on Low Power Electronics and Design*, 2002, pp. 172-177.

[2] Butts, M., A. DeHon and S. C. Goldstein, "Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips," *International Conference on Computer-Aided Design*, 2002, pp. 443-440.

[3] Melosh, N. A., et al., "Ultrahigh-Density Nanowire Lattices and Circuits," *Science*,Vol. 300, No. 5616, 2003, pp. 112-115.

[4] Whang, D., et al., "Large Scale Hierarchical Organization of Nanowire Arrays for Integrated Nanosystems," *Nano Letter*,Vol. 3, No. 9, 2003, pp. 1255-1259.

[5] Mishra, M. and S. Goldstein, "Scalable Defect Tolerance for Molecular Electronics," *Proceedings of the 1st Workshop on Non-Silicon Computing*, 2002, pp. 78-85.

[6] Han, J. and P. Jonker, "A defect- and fault-tolerant architecture for nanocomputers," *Nanotechnology*,Vol. 14, No. 2, 2003, pp. 224-230.

[7] Dehon, A. and H. Naeimi, "Seven Strategies for Tolerating Highly Defective Fabrication," *IEEE Design & Test of Computers*,Vol. 22, No. 4, 2005, pp. 306-315.

[8] Jeffery, C. M. and R. J. O. Figueiredo, "Hierarchical Fault Tolerance for Nanoscale Memories", *IEEE Transactions on Nanotechnology*,Vol. 5, No. 4, 2006, pp.407-414.

[9] Kleinosowski, A. J., et al., "Exploring Fine-Grained Fault Tolerance for Nanotechnology Devices with Recursive NanoBox Processor Grid,", *IEEE Transactions on Nanotechnology*,Vol.5, No. 5, 2006, pp. 575-586.

[10] DeHon, A. and M. J. Wilson, "Nanowire-Based Sublithographic Programmable Logic Arrays," *Proc. International Symposium on Field-Programmable Gate Arrays* , 2004, pp. 123-132.

[11] Goldstein, S. C., M. Budiu, "NanoFabrics: Spatial Computing using Molecular Electronics" *Proceedings of International Symposium on Computer Architecture*,2001, pp. 178-191.

[12] Iijima, S., "Helical Microtubules of Graphitic Carbon," *Nature*, vol 345, No.56, 1991, pp.56-58.

[13] Lyshevski, M. A., "Carbon Nanotubes Analysis, Classification and Characterization," *Proc. IEEE Conference on Nanotechnology*, 2004, pp. 527-529.

[14] Dresselhaus, M.S., G. Dresselhaus, and P. C. Eklund, *Science of Fullerenes and Carbon Nanotubes*, New York, NY: Academic Press, 1996.

[15] Raja, T., V. D. Agrawal, M. L. Bushnell, "A Tutorial on the Emerging Nanotechnology Devices," *International Conf. VLSI Design*, 2004, pp. 343-360.

[16] Dai, H., N. Franklin, and J. Han, "Exploiting the Properties of Carbon Nanotubes for Nanolithography," *Appl. Phys. Lett.*,Vol. 73, No. 11, 1998, pp. 1508-1510.

[17] Dai, H., et al., "Nanotubes as Nanoprobes in Scanning Probe Microscopy," *Nature*,Vol. 384, No. 6605, 1996, pp. 147-150.

[18] Wong, S. S., et al., "Covalently Functionalized Nanotubes as Nanometre-Sized Probes in Chemistry and Biology," *Nature* ,Vol. 394, No. 6688, 1998, pp. 52-55.

[19] Fuhrer, M. S., et al., "Crossed Nanotube Junctions," *Science*,Vol. 288, No. 5465, 2000, pp. 494-497.

[20] Bachtold, A., et al., “Logic Circuits with Carbon Nanotube Transistors,” *Science*,Vol. 294, No. 5545, 2001, pp. 1317-1320.

[21] Rueckes, T., et al., “Carbon Nanotube-Based Nonvolatile Random Access Memory for Molecular Computing”, *Science*,Vol. 289, No. 5476, 2000, pp. 94-97.

[22] Che, G., et al., “Carbon Nanotube Membranes for Electrochemical Energy Storage and Production,” *Nature*,Vol. 393, 1998, pp. 346-349.

[23] Derycke, V., et al., “Carbon Nanotube Inter- and Intramolecular Logic Gates,” *Nano Letters*,Vol. 1, No. 9, 2001, pp. 453-456.

[24] Javey, A., et al., “Ballistic Carbon Nanotube Field-effect Transistors,” *Nature*,Vol. 424, No. 6949, 2003, pp. 654-657.

[25] Huang, Y., et al., “Logic Gates and Computation from Assembled Nanowire Building Blocks,” *Science*,Vol. 294, No. 5545, 2001, pp. 1313-1317.

[26] Cui, Y. and C. M. Lieber, “Functional Nanoscale Electronic Devices Assembled Using Silicon Nanowire Building Blocks,” *Science*, Vol. 291, No. 5505, 2001, pp. 851-853.

[27] Beckman, R. A., et al., “Fabrication of Conducting Si Nanowire Arrays,” *Journal of Applied Physics*,Vol. 96, No. 10, 2004, pp. 5921-5923.

[28] Snider, G., P. Kuekes, R. S. Williams, “CMOS-like Logic in Defective, Nanoscale Crossbars,” *Nanotechnology*, vol 15, No. 8, 2004, pp. 881-891.

[29] Flood, A. H., et al., “Whence Molecular Electronics,” *Science*,Vol. 306, No. 5704, 2004, pp. 2055-2056.

[30] Heath, J. R. and M. A. Ratner, “Molecular Electronics,” *Physics Today*, 2003, pp. 43-49.

[31] Chen, Y., et al., “Nanoscale molecular-switch devices fabricated by imprint lithography,” *Applied Physics Letter*,Vol. 82, No. 10, 2003, pp. 1610-1612.

[32] Steuerman, D. W., et al., “Molecular-Mechanical Switch-Based Solid-State Electrochromic Devices,” *Angewandte Chemie International Edition*,Vol. 43, No. 47, 2004, pp. 6486-6491.

[33] Hadley, P., “Single-Electron Tunneling Devices,” *AIP conference proceedings 427*, 1998, pp.256-270.

[34] Averin, D. V. and K. K. Likharev, “Single electronics: a correlated transfer of single electrons and Cooper pairs in systems of small tunnel junctions,” *Mesoscopic phenomena in solids*,Vol. 30, pp. 173-271, B.L.Altshuler, P.A.Lee and R.A.Webb (eds), New Yok, NY: North-Holland, 1991.

[35] Likharev, K. K., “Single-Electron Devices and Their Applications,” *Proceedings of IEEE*,Vol. 87, No. 4, 1999, pp. 606-632.

[36] Lafarge, P., et al., “Direct observation of macroscopic charge quantization,” *Z. Phys. B*,Vol. 85, 1991, pp. 327-332.

[37] Fulton, T. A., P. L. Gammel, and L. N. Dunkleberger, “Determination of Coulomb-blockade resistances and observation of the tunneling of single electrons in small-tunnel-junction circuit,” *Phys. Rev. Lett.*,Vol. 67, 1991, pp. 3148-3151.

[38] Averin, D. V. and K. K. Likharev, “Possible Applications of the Single Charge Tunneling,” *Single Charge Tunneling*, pp. 311-322, H. Grabert and M.H. Devoret(eds), New York, NY: Plenum, 1992.

[39] Dresselhaus, P., et al., “Measurement of single electron lifetimes in a multijunction trap,” *Phys. Rev. Lett.*,Vol. 72, No. 20, 1994, pp. 3226-3229.

[40] Ji, L., et al., “Fabrication and characterization of single-electron transistors and traps,” *J. Vac. Sci. Technol. B*,Vol. 12, No. 6, 1994, pp. 3619-3622.

[41] Likharev, K. K., “SET: Coulomb Blockade Devices,” *Nano et Micro Technologies*,Vol. 3, No. 1-2, 2003, pp. 71-114.

[42] Geerligs, L. J., et al., “Frequency-locked turnstile device for single electrons,” *Phys. Rev. Lett.*,Vol. 64, No. 22, 1990, pp. 2691-2694.

[43] Pothier, H., et al., “Single electron pump fabrication with ultrasmall normal tunnel junctions,” *Physica B*,Vol. 169, 1991, pp. 1598-574.

[44] Chen, R. H., A. N. Korotov, K. K. Likharev,“Single electron transistor logic,” *Appl. Phys. Lett.*,Vol. 68, No. 14, 1996, pp. 1954-1956.

[45] Geppert, L.,“Quantum transistors: toward nanoelectronics,” *IEEE Spectrum*, Vol. 37, No. 9, 2000, pp. 46-51.

[46] Takahashi, Y., et al., “Silicon Single-Electron Devices and Their Applications,” *30th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2000)*, 2000, pp. 441-420.

[47] Mahapatra, S. and A.M. Ionescu,“A Novel Single Electron SRAM Architecture,” *IEEE Conference on Nanotechnology*, 2004, pp. 287-289.

[48] Inokawa, H., et al.,“A Multiple-valued Logic with Merged Single-electron and MOS Transistors,”*IEDM Tech. Dig.*, 2001, pp. 147-150.

[49] Takahashi, Y., et al., “Multigate Single-electron Transistors and Their Application to an Exclusive-OR Gate,” *Appl. Phys. Lett.*,Vol. 76, No. 5, 2000, pp. 637-639.

[50] Matsumoto, K., “Defective Carbon Nanotube Channel Single Electron Transistor With Ultra-High Coulomb Energy of 5000k,” *Keynote TNT2003*, 2003.

[51] Saitoh, M., H. Harata, and T. Hiramoto, “Room-Temperature Demonstration of Integrated Silicon Single-Electron Transistor Circuits for Current Switching and Analog Pattern Matching,” *IEEE Electron Devices Meeting (IEDM)*, 2004, pp. 187-190.

[52] Soldatov, E.S., et al., “Room Temperature Molecular Single-Electron Transistor,” *Phys. Usp.*, Vol. 41, No. 2, 1998, pp. 202-204.

[53] Uchida, K., et al., “Programmable single-electron transistor logic for future low-power intelligent LSI: Proposal and room-temperature operation,” *IEEE Transaction on Electron Devices*, Vol. 50, No. 7, 2003, pp. 1623-1630.

[54] Chang, L. L., L. Esaki, and R. Tsu, “Resonant tunneling in semiconductor double barriers,” *Appl. Phys. Lett.*, Vol. 24, No. 12, 1974, pp. 593.

[55] Liu, H. C. and T. C. L. G. Sollner, “High-frequency Resonant Tunneling Devices,” *High-Speed Heterostructure Devices, Semiconductors and Semimetals series*, Vol. 41, pp. 359-419, R.A. Kiehl and T.C.L.G. Sollner (eds), New York, NY: Academic Press, 1994.

[56] Özbay, E., et al., “1.7 ps, microwave, integrated-circuit-compatible InAs/AlSb resonant tunneling diode,” *IEEE Electron Device Lett.*,Vol. 14, No.8, 1993, pp. 400-402.

[57] Mathews, R. H., et, al., “A New RTD-FET Logic Family,” *Proc. IEEE*,Vol. 87, No. 4, 1999, pp. 596-605.

[58] Mazumder, P., S. Kulkarni, M. Bhattacharya, J. P. Sun, and G. I. Haddad, “Digital Circuit Applications of Resonant Tunneling Device,” *Proc. IEEE*, Vol. 86, No. 4, 1998, pp. 664-686.

[59] Goldstein, S.C. and D. Rosewater, “Digital Logic Using Molecular Electronics,” *IEEE International Solid-State Circuits Conference*, 2002, pp. 204-205.

[60] Seabaugh, A. C., J. H. Luscombe and J. N. Randall, “Quantum Functional Devices: Present Status and Future Prospects,” *Journal of Future Electron Devices (FED)*, col.3, suppl. 1, 1993, pp. 9-20.

[61] “International Technology Roadmap for Semiconductors,” *Jointly Sponsored by European Semiconductor Industry Assc.,Japan Electronics and Information Technology Industry Assc., Korea Semiconductor Industry Assc., Taiwan Semiconductor Industry Assc., and Semiconductor Industry Assc.*, 2004.

[62] Pramanik, S., S. Bandyopadhyay, M. Cahay, “Why is the Spin Field Effect Transistor Elusive?” *Proc. IEEE Conference on Nanotechnology*, 2004, pp. 101-103.

[63] Chen,Y., et al., “Nanoscale Molecular-Switch Crossbar Circuits,” *Nanotechnology*,Vol. 14, 2003, pp. 462-468.

[64] Ziegler, M. M. and M. R. Stan, “Design and Analysis of Crossbar Circuits for Molecular Nanoelectronics,” *IEEE International Conference on Nanotechnology*, 2002, pp. 323-327.

[65] Reed, M. A., et al., “The Design and Measurement of Molecular Electronic Switches and Memories,” *ISSCC Dig. Tech. Papers*, 2001, pp. 114-115.

[66] DeHon, A., P. Lincoln, J. E. Savage, “Stochastic Assembly of Sublithographic Nanoscale Interfaces,” *IEEE Trans. on Nanotechnology*,Vol. 2, No. 3, 2003, pp. 165-174.

[67] Dehon, A., “Design of Programmable Interconnect for Sublithographic Programmable Logic Arrays,” *Proc. International Symposium on Field-Programmable Gate Arrays* , 2005, pp. 127-137.

[68] Gojman, B., et al., “3D Nanowire-Based Programmable Logic”, *Proceedings of International Conference on Nano-Networks*, 2006.

Chapter 3

QCA

M. Momenzadeh, J. Huang, and F. Lombardi

QCA is a novel emerging technology in which logic states are not stored as voltage levels, but rather the position of individual electrons. Conceptually, QCA represents binary information by utilizing a bistable charge configuration rather than a current switch. A QCA cell can be viewed as a set of four "dots" that are positioned at the corners of a square. A quantum dot is a site in a cell in which a charge can be localized. The cell contains two extra mobile electrons that can quantum mechanically tunnel between dots, but not cells. In the ground state and in the absence of external electrostatic perturbation [1], the electrons are forced to the corner positions to maximize their separation due to Coulomb repulsion. As shown in Figure 3.1, the two possible charge configurations are used to represent binary "0" and "1". Note that in the case of an isolated cell, the two polarization states are energetically degenerate. However the presence of other charges (neighbor cells) breaks the degeneracy and one polarization state becomes the cell ground state [1]. Polarization P measures the extent to which the charge distribution is aligned along one of the diagonal axes. If the charge density on a dot i is ρ_i, then the polarization is defined as [2] [3]:

$$P = \frac{(\rho_1 + \rho_3) - (\rho_2 + \rho_4)}{\rho_1 + \rho_2 + \rho_3 + \rho_4} \tag{3.1}$$

The tunneling between dots implies that ρ_i may not be integers as polarization values.

Figure 3.2 illustrates the cell-to-cell response function, in which the polarization P2 of cell 2 is induced by the fixed polarization of a driver (i.e., its neighbor,

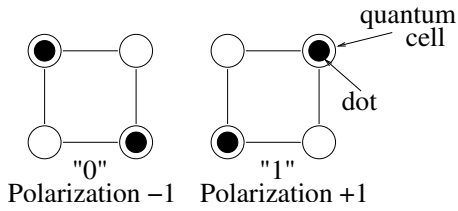


Figure 3.1 QCA Cell

or cell 1 in this case) [3]. In the ground state of this two-cell system (that corresponds to a correct computation), the polarization P2 is aligned with its neighbor polarization P1. The cell-cell response curve can be computed by solving the two particle Schrodinger equation [1]. It can been seen that the cell-cell response is highly non-linear, which indicates signal restoration. Even a slightly polarized input cell induces an almost fully polarized output cell.

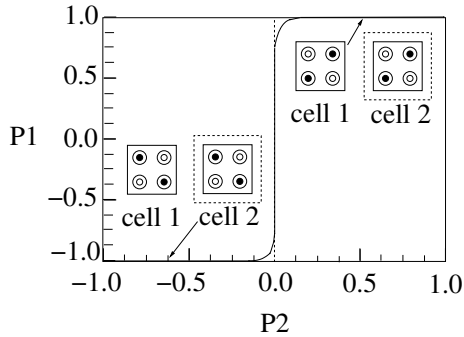


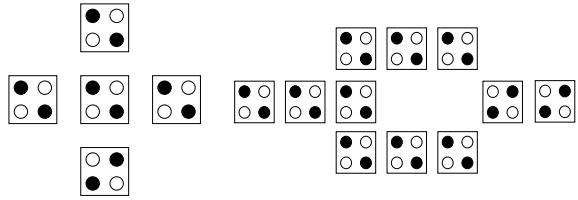
Figure 3.2 QCA Polarization States

A driver of a QCA cell could be an input device such as a nanotube, a very thin wire or a tip of a scanning tunneling microscope (STM). In semiconductor QCA, a standard technique called “plunger electrode” has been used to alter the electron occupancy of the input cell [4] [5] [6]. Reading the output state of a QCA cell is difficult, because the required measurement process must not change the charge of the output cell. Electrometers made from ballistic point-contacts [7] [8], the STM method [9], and SET electrometer have been used to read the output.

Unlike conventional logic circuits in which information is transferred by electrical current, QCA operates by the Coulombic interaction that connects the state of

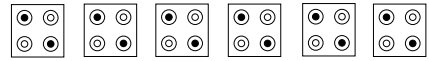
one cell to the state of its neighbors. For QCA, this results in a technology in which information transfer (interconnection) is the same as information transformation (logic manipulation).

Various types of QCA devices can be constructed using different physical cell arrangements. One of the basic logic gates in QCA is the majority voter (MV) with logic function $MV(A, B, C) = AB + AC + BC$. MV can be realized by 5 QCA cells, as shown in Figure 3.3(a). Logic AND and OR functions can be implemented from the MV by setting an input (the so-called programming or control input) permanently to a "0" or "1" value. The inverter (INV) is the other basic gate in QCA and is shown in Figure 3.3(b). In INV, the 45^o displacement in the two lines of merging cells, produces complement action of the input signal. Unlike conventional CMOS in which it is the simplest block, the inverter consumes a substantial area in QCA.

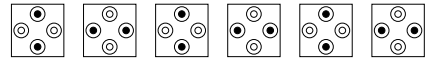


(a) Majority Voter (b) Inverter

Figure 3.3 Basic QCA Devices



(a) Binary Wire



(b) Inverter Chain

Figure 3.4 QCA Interconnects

The binary wire and inverter chain (as interconnect fabric) are shown in Figure 3.4(a) and (b) respectively. In the binary wire, a signal propagates from the input to the output. Due to the presence of 45^o (rotated) cells in an inverter chain, the signal

alternates between the input value and its logic complement as it traverses the chain towards the output. By connecting a 90^o (non-rotated) cell in the middle of two of these 45^o (rotated) cells (as shown in Figure 3.5), both the original input signal and its complement can be obtained.

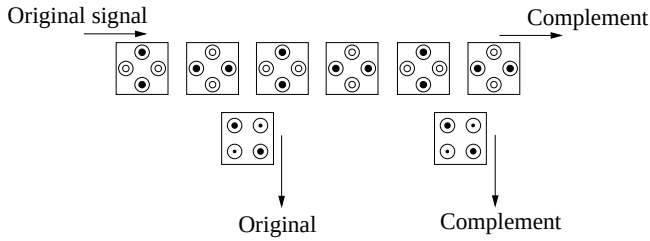


Figure 3.5 Inverter Chain, Original and Complement Signal

Crossing of two wires in one plane is achieved by placing a binary wire (90^o) between two inverter chains (45^o) as shown in Figure 3.6. The two signals are able to cross each other without interference since the wires of different orientation do not have any switch effect on each other [3].

A QCA circuit consists of an array of QCA cells arranged in a Cartesian plane. QCA computes by mapping the energy ground state of the QCA array to the logic solution of the problem. The input cells of the QCA array are in fixed polarization, the entire array is then allowed to relax to its ground state. The output is read by sensing the state of the output cells. What distinguishes input cells from output cells is the fact that input cells are held in fixed polarization while the output cells are allowed to switch to whatever polarization that achieves system ground state [1]. QCA system computes correctly when the array settles to its ground state. When the system is stuck in a metastable state (no the true energy ground state), a *kink* occurs. The *kink energy* E_k is the energy required to excite the system from the ground state to the first excited state. To distinguish a bit value from the thermal environment E_k must be greater than k_BT [10] where T is the operation temperature in degrees Kelvin and k_B is Boltzmann's constant. It has been proved in [1] that to avoid kink the number of QCA cells N in the longest line must be less than $e^{\frac{E_k}{k_BT}}$. If the ratio E_k/k_BT is 4, N is about 50; however if E_k/k_BT is increased to 10 (by either lowering temperature or raising E_k), N exceeds 22,000.

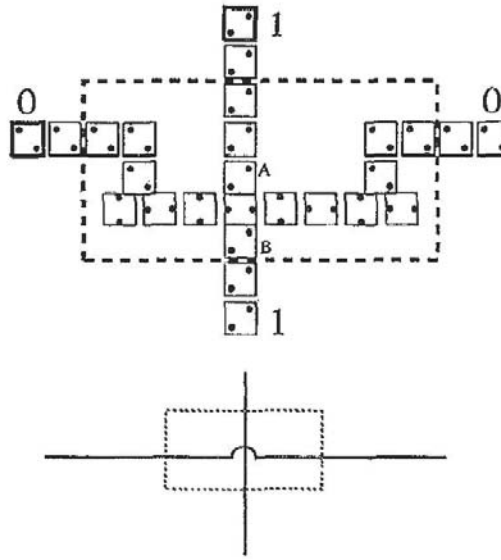


Figure 3.6 Coplanar Wire Crossing of Two QCA Wires (From [3]. ©1994 Journal of Applied Physics. Reprint with permission)

3.1 QCA IMPLEMENTATION

There have been several proposals for physically implementing QCA: Micro-sized QCA devices have been fabricated with metal which operate at $50\,mK$ [11] [12] and an extensive literature has been reported on developing molecular implementations of QCA [13] [9]. Magnetic QCA (MQCA) has been investigated and fabricated [14] [15] for room temperature operation. In this section, a brief background on Metal, Molecular, and Magnetic QCA is provided.

3.1.1 Metal QCA

In [12], an experimental demonstration of a basic QCA cell has been presented. This device is composed of four aluminum islands (as dots) connected with aluminum-oxide tunnel junctions and capacitors. The area of the tunnel junctions determines the island capacitance (the charging energy of the dots) and hence, the operating temperature of the device. The device has an area of approximately $60 \times 60nm^2$ and is mounted on a surface at 10mK temperature. The device has been fabricated using Electron Beam Lithography (EBL) and dual shadow evaporation on an oxidized silicon wafer [11]. The simplified schematic diagram of this cell is shown in Figure 3.7. The aluminum dots are located at D1 through D4, coupled by tunnel junctions. The two dots (E1 and E2) are SET electrometers for sensing the output. Figure 3.8 shows the scanning electron micrograph of this QCA cell.

Experiments have confirmed that switching of electrons in a cell can control the position of electrons in another cell. In [16], basic logic circuits made of these cells have been demonstrated. Sequential circuits have also been fabricated using metal tunnel junction technology; the operation of a QCA latch and a two-bit shift register have been demonstrated in [17] [18] and implemented in [19]. Figure 3.9 illustrates the schematic and electrical diagrams of a QCA latch. This device consists of three floating micron-size metal dots (D_1-D_3), connected in series by multiple tunnel junctions (MTJ) and controlled by capacitively coupled gates. The electrometer (E_1), the signals ($-V_{IN}, +V_{IN}$), and the clock (V_C) are coupled to the dots. The operating temperature of this devices is 70mK. It is predicted that molecular scale ($\sim 2nm$) will yield room temperature for QCA.

A semiconductor implementation of QCA is advantageous due to well understood behavior of existing semiconductors for which several tools and techniques have been already developed [20]. However, fabrication processes are not suitable to mass produce QCA cells of sufficiently small dimensions (few nanometers) for operating at room temperature.

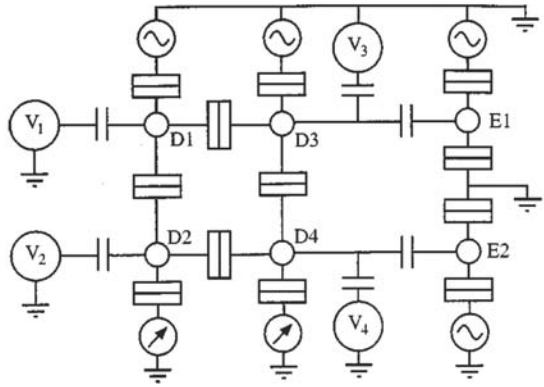


Figure 3.7 Simplified Schematic Diagram of Four-dot Metal QCA Cell (From [7]. ©1999 Nanotechnology. Reprint with permission)

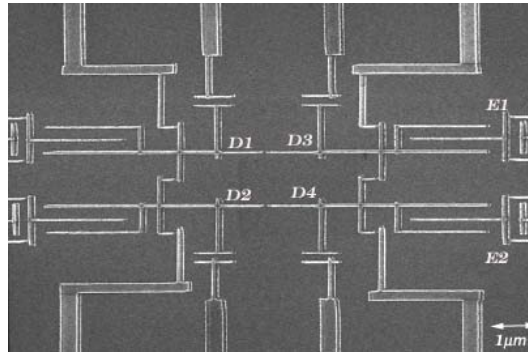


Figure 3.8 Scanning Electron Micrograph of the QCA Device [7]

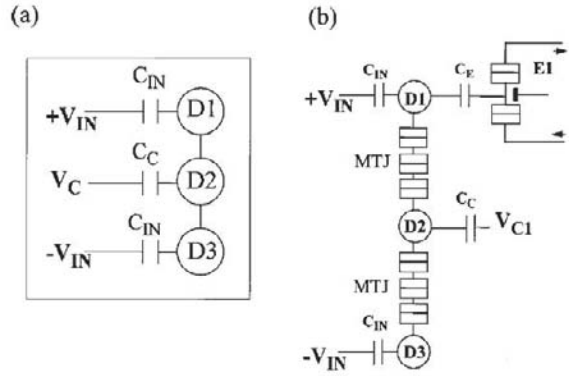


Figure 3.9 a) Schematic and b) Electrical Diagrams of Half Cell QCA Latch (From [19]. ©2001 Applied Physics Letters. Reprint with permission)

3.1.2 Molecular QCA

As an alternative technology, molecular QCA has several advantages over metal-dot QCA; small cell size (density of up to 10^{13} devices per cm^2), a simple manufacturing process, and operation at room temperature are some of the desirable features of molecular QCA. Moreover, an improvement of switching speed by 100 times in molecular-sized QCA cells has been reported over semiconductor QCA cells [21]. A further advantage of molecular QCA is that cells are structurally homogeneous down to the atomic level [22]. It has been shown that mix-valence complexes can be used to construct QCA cells [23] [24]. An initial analysis of a simple molecular system that operates as a molecular QCA cell has been presented in [23]; each molecule functions as a QCA cell and redox centers act as "quantum dots" in which information is encoded with charge configurations and tunneling is provided by bridging ligands.

Recent experiments suggest to use nonbonding orbitals (π or d) as dot sites for a QCA molecule. Two, three, or four dot molecules have been fabricated. For example, the Trans-Ru(dppm)$_2$(C$\equiv$CFc)(NCCH$_2$CH$_2$NH$_2$) dication is a two redox center molecule that has been synthesized and attached on a silicon substrate [24]. The quantum dots in the molecules are ferrocence and Ru(dppm)$_2$ groups, while the tunneling junction for the mobile electron is provided by the carbon-carbon triple bond. Two molecules form a four-dot QCA cell.

Another recent experiment has synthesized the $\{\eta 5\text{-}C_5H_5\}Fe(\eta 5\text{-}C_5H_4)\}_4(\eta 4\text{-}C_4)Co(\eta 5\text{-}C_5H_5)$ dication as a four redox center [24]. Two mobile electrons can tunnel through the $(\eta 4\text{-}C_4)Co(\eta 5\text{-}C_5H_5)$ group. A theoretical demonstration of these two QCA molecules has been presented in [24].

Molecular QCA presents unique challenges: bonding of the array surface requires complexes by spectroscopic and electrochemical techniques [25]. Moreover, the presence of strongly bound, chemically robust, mixed valence complexes in the required chemistry has been extensively treated. Perturbation of the chemical complex by surface binding using a gold electrode by an electrochemical method has been investigated resulting in an assembly of biased, vertically oriented two-dot structures (dipole) sandwiched between two electrodes [25]. The assembly of a symmetric square cell (containing two ferrocene and two ferrocenium moieties) with measured properties that make it suitable as a component for charge-coupled QCA circuits, has been shown in [26]. However, deposition defects are still widely reported and they must be carefully considered because they may affect the correct operation of QCA circuits. It will be shown in Chapter 5.2 that cell deposition defects create unwanted cell interaction and thus logic level errors. Another challenge in molecular QCA (in addition to deposition defects) is the I/O interface which must be provided with a single molecule.

3.1.3 Magnetic QCA

Cowburn and Welland have proposed a magnetic implementation of QCA (MQCA) in 2000 [14]. In MQCA, magnetostatic interactions between nanoparticles ensure that the system is bistable. The moments of the nanoparticles point either parallel, or anti-parallel with the axis of the chain, as shown in Figure 3.10. Information is propagated via magnetic exchange interactions as opposed to the electrostatic interactions in metal and molecular implementations.

Figure 3.10 Vector Magnetization in MQCA (From [15]. ©2003 M. Parish. Reprint with permission)

Cowburn and Welland have demonstrated experimentally [14] that MQCA using relatively large dots (about $100nm$ in size) operates at room temperature. MQCA provides the advantage of operation at room temperature even with current

fabrication techniques. However, magnetic QCA does not appear to have the switching speed to compete with today's computers (such as an alternative for designing memories) [27].

3.2 CLOCKING

In VLSI systems, timing is controlled through a reference signal (i.e., a clock) and is mostly required for sequential circuits. Timing in QCA is accomplished by clocking in four distinct and periodic phases [28] and is needed for both combinational and sequential circuits. Clocking provides not only control of information flow but also true power gain in QCA [29]. Signal energy lost to the environment is restored by the clock.

Two types of switching methods are possible in the operation of QCA: abrupt switching and adiabatic switching. In abrupt switching, the inputs to the QCA circuit change suddenly and the circuit can be in some excited state; subsequently, the QCA circuit is relaxed to ground state by dissipating energy to the environment [30]. This inelastic relaxation is uncontrolled and the QCA circuit may enter a metastable state that is determined by a local, rather than a global energy ground state. Therefore, adiabatic switching is usually preferred; in adiabatic switching, the system is always kept in its instantaneous ground state. A clock signal is introduced to ensure adiabatic switching.

For QCA, the clock signals are generated through an electric field, which is applied to the cells to either raise or lower the tunneling barrier between dots within a QCA cell. This electric field can be supplied by CMOS wires, or CNTs [31] buried under the QCA circuitry. When the barrier is low, the cells are in a non-polarized state; when the barrier is high, the cells are not allowed to change state. Adiabatic switching is achieved by lowering the barrier, removing the previous input, applying the current input and then raising the barrier [30]. If transitions are gradual, the QCA system will remain close to the ground state.

The clocked QCA circuit utilizes the tri-state six-dot cells, as shown in Figure 3.11. The clock signal is applied to either push the electrons to the four corner dots or pull them into the two middle dots. When the electrons are in the middle dots, the cell is in the "null" state. When the electrons are in the four corner dots, the cell is in an active state. The charge configuration of the cell in active state represents binary "0" and "1" as shown previously in Figure 3.1. A molecule with three quantum-dot hole sites is shown in Figure 3.12. Two such molecules form a six-dot QCA cell [29].

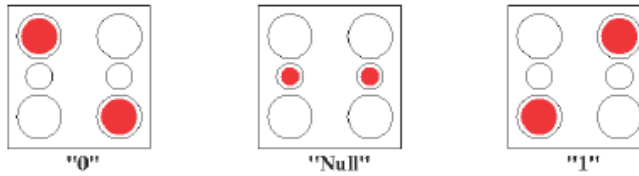


Figure 3.11 Schematic Diagram of a Six-dot QCA Cell

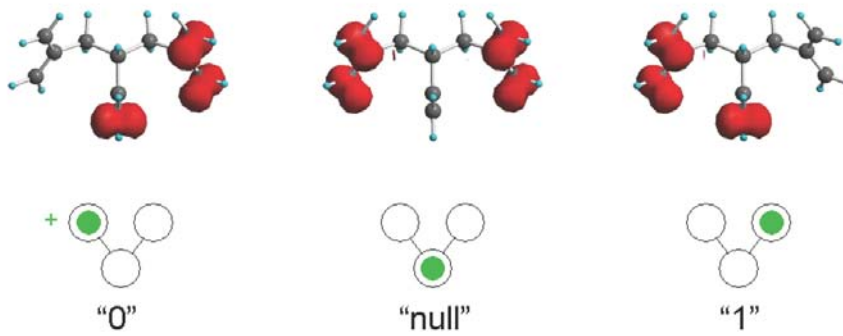
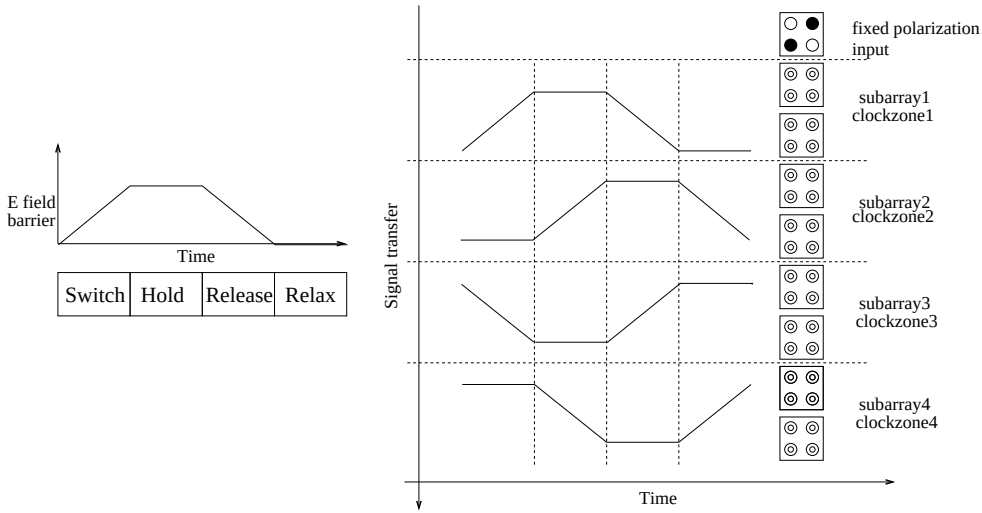


Figure 3.12 Tri-state Quantum-dot Molecule (From [32]. ©2003 IEEE. Reprint with permission)



(a) 4 Phase Clocking

(b) Switching of a Binary Wire

Figure 3.13 Clocking in QCA

This clocking scheme (which was introduced in [28]) consists of four phases: Switch, Hold, Release and Relax, as shown in Figure 3.13(a). The QCA circuit is partitioned into so-called clocking zones, such that all cells in a zone are controlled by the same clock signal. Cells in each zone perform a specific calculation. During the Relax phase, the electrons are pulled into the middle dots, so the cell is in "null" state. During the Switch phase, the interdot barrier is slowly raised and pushes the electrons into the corner dots, so the cell attains a definitive polarity under the influence of its neighbors (which are in the Hold phase). In the Hold phase, barriers are high and a cell retains its polarity and acts as input to the neighboring cells. Finally in the Release phase, barriers are lowered and the electrons are pulled into the middle dots so the cell loses its polarity. Here switching is adiabatic, i.e. the system remains very close to the energy ground state during transition, and the stationary state of each cell can be obtained by solving the time-independent Schrodinger equation. Clocking zones of a QCA circuit or system are arranged in this periodic fashion, such that zones in the Hold phase are followed by zones in the Switch, Release and Relax phases. A signal is effectively "latched" when one clocking zone goes into the Hold phase and acts as input to the subsequent zone.

In a clocked QCA circuit, information is transferred and processed in a pipelined fashion [33] [20] and allows multi-bit information transfer for QCA through signal latching. All cells within the same zone are allowed to switch simultaneously, while cells in different zones are isolated. Consider the binary wire in Figure 3.13(b); initially, $subarray1$ switches according to the fixed input, and $subarray2$ shows no definite polarization at this time. Then, $subarray1$ enters the Hold phase; at this time $subarray2$ starts switching. As $subarray3$ is in the Relaxed state, it will not influence the computational state of $subarray2$. Next, $subarray1$ is moved to a Release phase; $subarray2$ is in the Hold state and serves as the input to $subarray3$ (which is in the switch phase). The signal is "latched" when $subarray1$ enters the Hold phase and acts as input to $subarray2$.

In the adiabatic switching schemes, fluctuations in operating temperature T may excite QCA cells above their ground state and produce erroneous results at the output. An analysis of these thermal effects on a line of N QCA cells is provided in [1]. It has been shown in [1] that for reliable kink-free computation, within a single clocking zone, N is bound by $e^{\frac{E_k}{k_B T}}$. Large QCA circuits are therefore partitioned into smaller subcircuits, each of which resides in its own clocking zone.

The clock signal is commonly generated by CMOS wires buried under the QCA circuitry. Figure 3.14 depicts the schematic diagram for clocking a 3-dot molecular QCA array [28]. QCA molecules are located in the xz plane and clock wires are placed in the z direction, thus inducing an electrical field in the y direction.

One of the limiting factors for high density of QCA systems is the wiring requirements for the generation of the electrical field. The use of single walled carbon nanotubes (SWNTs) and a new clock wire layout is recommended in [31]. It has been shown that metallic SWNTs are excellent conductors [34] and can be used to generate a clocking field that smoothly propagates the QCA signals. The layout method of [35] consists of a series of clocking wires perpendicular to the QCA signal direction, as shown in Figure 3.15(a). In this method, the direction of the perpendicular clocking wires must be changed with turns in the QCA signals (as shown in Figure 3.15(b)). The approach proposed in [31] allocates clocking wires at a 45^o angle (Figure 3.15(c)); hence, only two clocking directions (perpendicular to each other) are needed to allow QCA signal propagating along the two axes.

3.3 MOLECULAR ATTACHMENT

Matching the pitch between cells and the substrate on which they are attached [9] [36], is a significant issue for QCA. Currently, top-down lithography methods do not

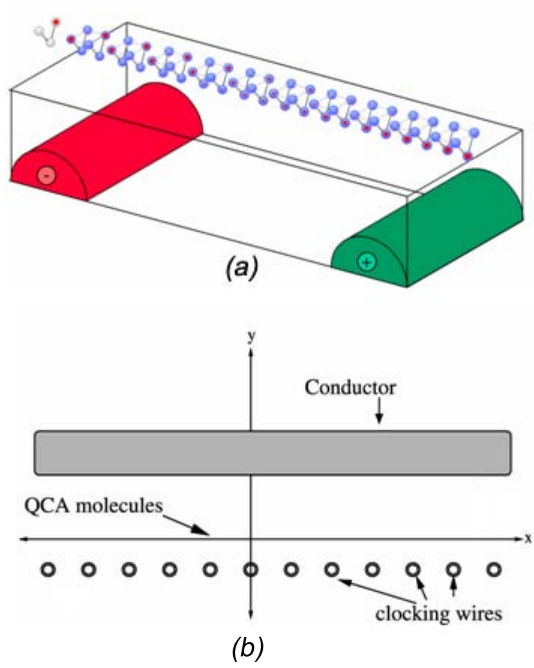


Figure 3.14 Clocking Schematic Diagram of a Molecular QCA Array (From [32]. ©2003 IEEE. Reprint with permission)

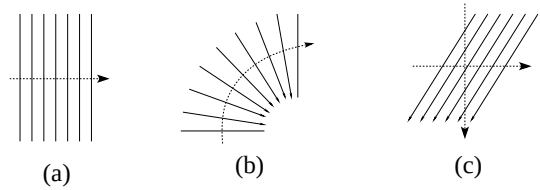


Figure 3.15 Clocking Layout (From [31]. ©2004 IEEE. Reprint with permission)

meet the demand for generating detailed attachment patterns due to their limitations in high resolution and throughput. An an alternative process to the expensive nanoscale lithography is to use a self-assembly method, as a bottom-up approach. This method has been previously used for creating patterns of nanoparticles and nanocrystals at molecular scale [37] [38] [39] [40]. However, manufacturing of bottom-up assembled QCA systems is very challenging. As an alternative solution to this problem, a methodology based on a combination of top-down lithography and the Self-Assembly Monolayer (SAM) method has been proposed in [41]. A molecular QCA circuit is constructed by allowing self-assembly of QCA cells on DNA rafts and using Electron Beam Lithography (EBL) to position the DNA rafts into trenches. DNA tiles are utilized, because they can form stable and well defined patterns and later, assemble into complex combinations by self-assembly [42]. Each DNA raft contains a number of tiles; each tile can hold several QCA cells. Lieberman et. al. have synthesized four-tile DNA rafts in which each tile contains eight QCA cells [43] [44]. Since EBL is not capable by itself of defining patterns below $10nm$, a cold-development technique has been used in [43] to reduce the patterning resolution to $5nm$. The manufacturing process of a layout is shown in Figure 3.16; tiles made of double helices are assembled to generate a substrate on which the QCA cells are deposited [45]. Different layouts can be made by combining the possible configurations of eight-cell tiles which hold QCA cells in fixed positions.

3.4 POWER GAIN AND DISSIPATION

Energy dissipation causes a signal to degrade from stage to stage through its propagation path and eventually, this may result in a signal loss in the thermal background. A power supply and transistors are utilized in conventional CMOS circuits to restore the energy lost to dissipative processes. In QCA circuits, energy is restored by the clocking process and related electric field; when the signal strength in a QCA cell is reduced, the electric field provides additional energy to deliver copies of the cell's signal to the neighboring cells, while clocking takes place.

Power gain for molecular QCA has been analyzed theoretically in [10] and experimentally measured for some metal-dot QCA devices in [46]. In [46] Kummamuru et. al. have evaluated the change in the signal power as it passes through a latch by measuring the average energy of the input and output signals over one clock cycle. The work performed by a latch over a given time interval (by

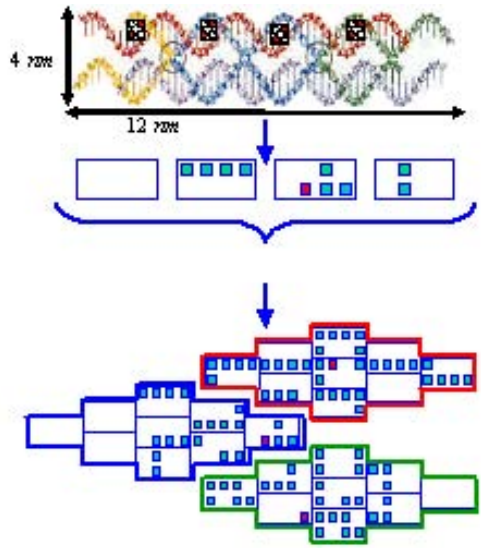


Figure 3.16 Layout Synthesis with DNA Tiles (From [45]. ©2004 IEEE. Reprint with permission)

a particular lead voltage) has been found as:

$$W = \int V dQ = \int_0^{t'} V_L(t) \frac{dQ_C(t)}{dt} dt$$

where $V_L(t)$ is the voltage applied to the lead, $Q_C(t)$ is the charge on the capacitor coupling the dot to the voltage lead, and t' is the given time interval. Power gain has been defined as the ratio of the output to input signal power:

$$Powergain = \frac{P_{out}}{P_{in}} = \frac{W_{out}/T}{W_{in}/T}$$

This ratio is 2.07 and 2 for the experimental and the theoretical results respectively in a metal-dot QCA latch [46].

Figure 3.17 [10] illustrates that ultralow power dissipation can be achieved at molecular QCA densities and the calculated power dissipation of molecular QCA compared to existing and projected technologies. The upper bound for the QCA region is the worst case scenario wherein all cells switch non-adiabatically, thus

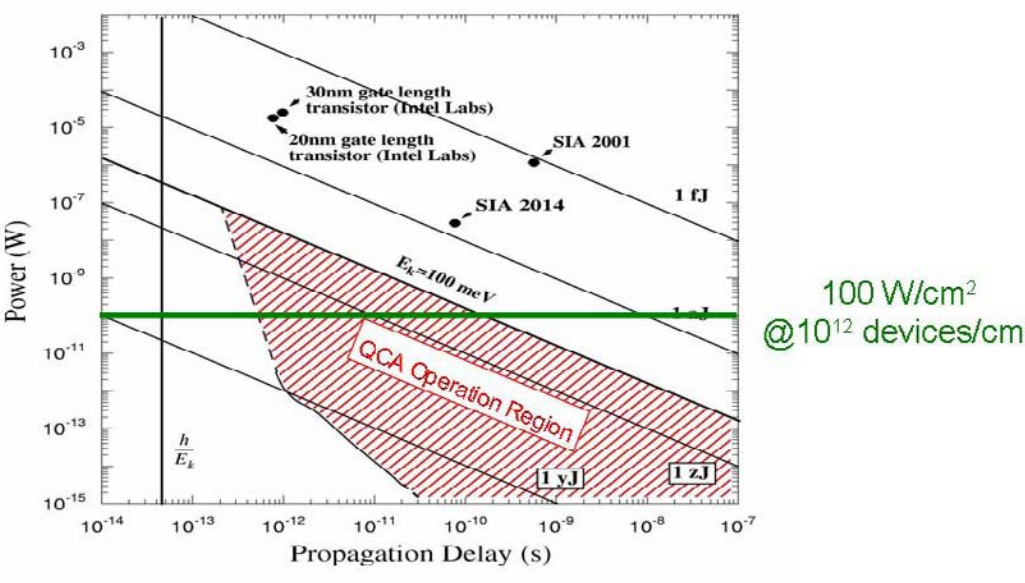


Figure 3.17 Power Comparison (From [10]. ©2002 Journal of Applied Physics. Reprint with permission)

resulting in dissipating the full value of the kink energy E_k (in this case 100 meV) for every clock cycle. The lower bound for the QCA region is the best case scenario wherein every cell switches quasi-adiabatically. The points labelled in the figure are for 2001 and 2014, as reported by the SIA roadmap [47] and transistors fabricated by Intel with 20 and $30nm$ gate length. Note that thermal noise and error correction are not included in the calculation of the energy dissipated.

Recently QCA has been advocated as a technology for reversible computing [48][49], in which virtually no dissipation scenario can be achieved. Reversible computing and QCA are explored in more detail in Chapter 11, using a QCA model proposed for QCA energy and dissipation analysis.

3.5 QCA SIMULATORS

Several QCA simulators are currently available [35] [50] [51] [53] [27]. mAQUINAS [35] and QCADesigner [27] are physics-based and solve quantum equations for

cell interactions. mAQUINAS assumes a continuous clocking scheme envisioned for the molecular QCA systems [35]. Adiabatic switching is assumed where the system is kept close to the ground state. At each time step, the time independent Schrodinger equation is solved for each cell. The process continues until a self-consistent solution is found for the entire system [35]. QCADesigner [27] has been used to produce results presented in the book, and will be discussed in more detail next. However, quantum simulation is computation intensive and are not suitable for large circuits. QBert [50] is another simulator developed for digital logic simulation for QCA which can be run much faster. A new model based on a SPICE model has been proposed and experimentally verified in [54] [53]. A standard spice simulator can then be used in simulating QCA circuits. HDLQ has been proposed in [51] as a Hardware Description Language (HDL) based design tool for QCA. An HDL model of QCA devices has been presented in [51], which allows the user to verify the logic characteristics of QCA system using the HDLQ environment. HDLQ is applicable to QCA circuits with novel timing scheme in which timing zones are not necessarily placed in a cascade (one-dimensional) arrangement.

3.5.1 QCADesigner

QCADesigner v1.4.0 (Unix version) has been used extensively this book [27]. The two simulation engines have been used in this book, namely the *Bistable Engine* and the *Coherence Vector Engine*. Their principles are briefly reviewed next.

3.5.1.1 Bistable Simulation Engine

In the bistable engine, each cell is modeled as a simple two-state system. The bistable engine utilizes an approximation based on the interaction between cells, namely the interaction strength between two cells decays inversely with the fifth power of the distance separating them. Hence using this engine not all cell effects are considered. Only cell effects within an area defined by the so-called radius of effect R are considered for each cell i. For cell i, its two-state system model is mathematically described by the following Hamiltonian:

$$H_i = \sum_j \begin{bmatrix} -\frac{1}{2}P_j E^k_{i,j} & -\gamma \\ -\gamma & \frac{1}{2}P_j E^k_{i,j} \end{bmatrix} \tag{3.2}$$

$E^k_{i,j}$ is the kink energy between the two cells (i and j), which represents the energy cost of opposite polarization in the two cells; P_j is the polarization of cell j and γ is the tunneling energy. For each cell i, the sum of the Hamiltonian is

over all cells (i.e., j) within its *radius of effect R*. Switching is assumed to be adiabatic (i.e., the system remains very close to the energy ground state during transition). Therefore, the stationary state of each cell can be obtained by solving the time-independent Schrodinger equation. The QCADesigner engine uses the Jacobi algorithm to find the eigenvalues and eigenvectors of the Hamiltonian. The engine computes the polarization of each cell until the whole system converges.

3.5.1.2 Coherence Vector Simulation Engine

QCADesigner v1.4.0 features also a new simulation engine, namely the coherence vector engine. The coherence vector engine is based on the density matrix approach [27] which models the power dissipative effects of QCA. Unlike the bistable engine, this engine performs a time-dependent simulation of the QCA design [10]. Again, each cell is modeled as a two-state system that is represented by the Hamiltonian of (3.2). The radius of effect R determines the operation of each cell.

The coherence vector λ is a vector representation of the density matrix ρ of a cell, projected onto the basis spanned by the Identity and the Pauli spin matrices σ_x, σ_y, σ_z. The components of λ can be found by taking the Trace of the density matrix and multiplying it by each of the Pauli spin matrices. The polarization of each cell i is the z component of the coherence vector. The vector Γ represents the energy environment of the cell, including the effect of neighboring cells; this vector is given by:

$$\vec{\Gamma} = \frac{1}{\hbar}[-2\gamma, 0, \sum P_j E^k_{i,j}] \tag{3.3}$$

The simulation engine evaluates the equation of motion; this is a partial differential equation with an explicit time marching algorithm. The effective neighborhood of cell i determines the summation index and the equation of motion for the coherence vector (inclusive of dissipative effects) is given by:

$$\frac{\partial}{\partial t}\vec{\lambda} = \vec{\Gamma} \times \vec{\lambda} - \frac{1}{\tau}(\vec{\lambda} - \vec{\lambda_{ss}}) \tag{3.4}$$

τ denotes the relaxation time and λ_{ss} is the steady state coherence vector which is given by:

$$\vec{\lambda_{ss}} = -\frac{\vec{\Gamma}}{|\vec{\Gamma}|}\tanh(\frac{\hbar|\vec{\Gamma}|}{2k_B T}) \tag{3.5}$$

For every time step, Γ and λ_{ss} of each cell are evaluated and the coherence vector for each cell is stepped forward in time.

3.6 QCA CIRCUITS

Inverters and MVs provide a functionally complete logic set for QCA. Various QCA circuits, including combinational as well as sequential circuits have been proposed in the literatures. These includes adders, shift registers, RAM and a simple microprocessor [55] [56] [30] [57] [3] [58].

The schematic diagram of a single bit full-adder [3] implemented with 5 majority voters and 3 inverters, is shown in Figure 3.18. A, B are the operand inputs and C_{i-1} is the carry from the previous stage. The sum and carry bits are denoted as the S and C_i outputs. Figure 3.19 illustrates the ground state charge distribution for the case in which logic "0" and logic "1"s are assigned to the carry-in and each input lines respectively.

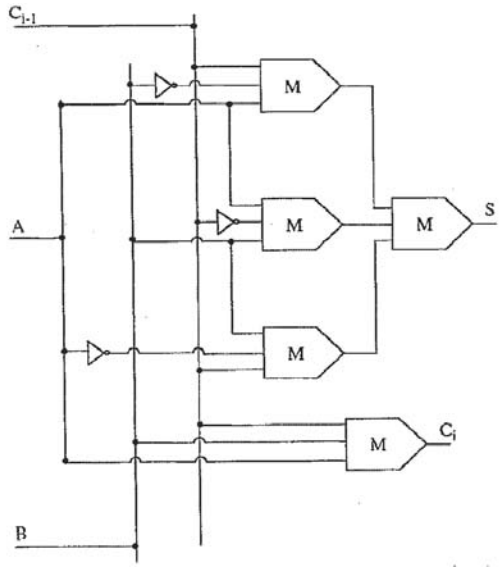


Figure 3.18 Schematic Diagram of Single-bit Full-Adder (From [3]. ©1994 Journal of Applied Physics. Reprint with permission)

Frost et. al. have presented an H-memory structure [56] that has the potential of dense storage with excellent processing capabilities. Figure 3.20 depicts the H-memory structure in QCA and its logic-level equivalent.

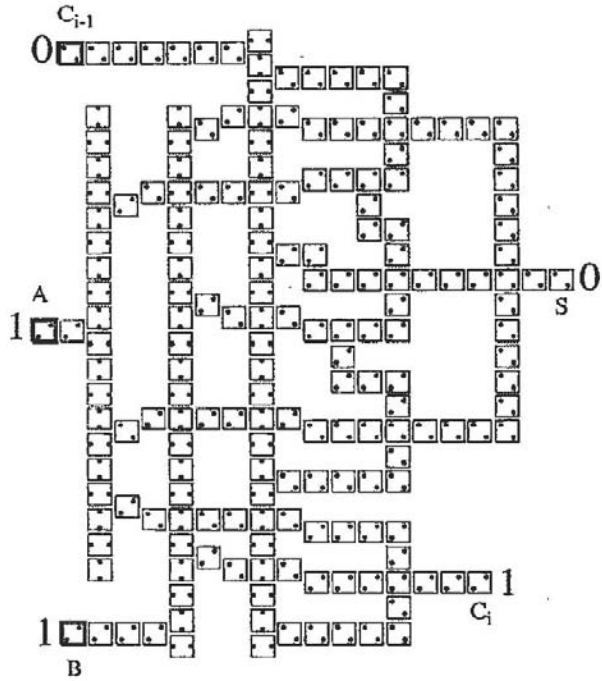


Figure 3.19 Layout of the Single-bit Full-Adder (From [3]. ©1994 Journal of Applied Physics. Reprint with permission)

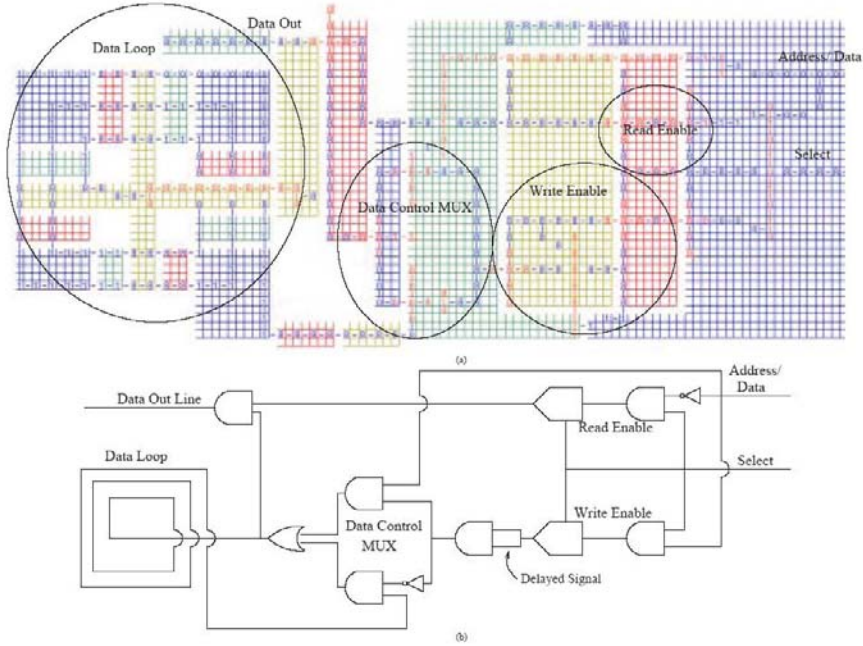


Figure 3.20 a) Layout of H-memory Structure b) Logic-level Equivalent Circuit (From [56]. ©2002 S.E. Frost. Reprint with permission)

A RAM design has been proposed and simulated in [58]. This RAM is based on a two-dimensional grid of memory cells and a scheme in which storage is kept in a circulating loop (Figure 3.21). A decoder (as shown in Figure 3.22) that generates a Select signal, is required to address any row of the QCA memory. Inverter chains are used for the control signals (S_0 and S_1) to provide the true and complement of these signals. Figure 3.23 shows a 1×4 RAM layout with a serial OR output array.

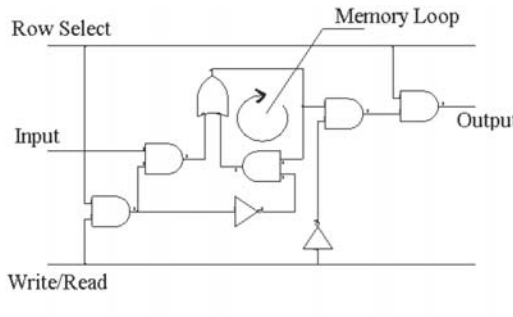


Figure 3.21 Schematic Diagram of QCA Memory Cell (From [58]. ©2003 Nanotechnology Conference. Reprint with permission)

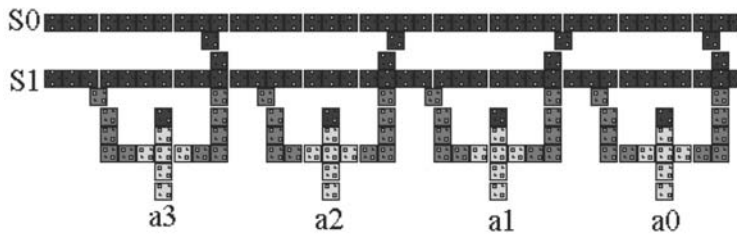


Figure 3.22 Decoder Layout (From [58]. ©2003 Nanotechnology Conference. Reprint with permission)

A simple ALU with a 12-bit data bus and an 8-bit addressable memory has been designed (as shown in Figure 3.24) by mapping the logic of its CMOS version to an equivalent QCA representation [57]. Due to problems arising from different clocking zone width, a large number of cells per clocking phase[1], and lack of

1 These two timing constraints will be discussed in Section 3.2.

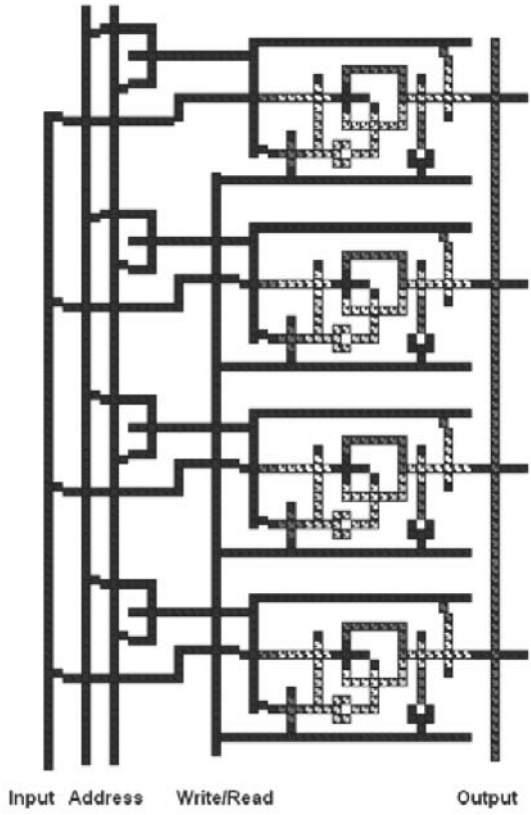


Figure 3.23 1×4 RAM Layout (From [58]. ©2003 Nanotechnology Conference. Reprint with permission)

physical feedback the original design was then modified [57]. Figure 3.25 illustrates a portion of the modified ALU.

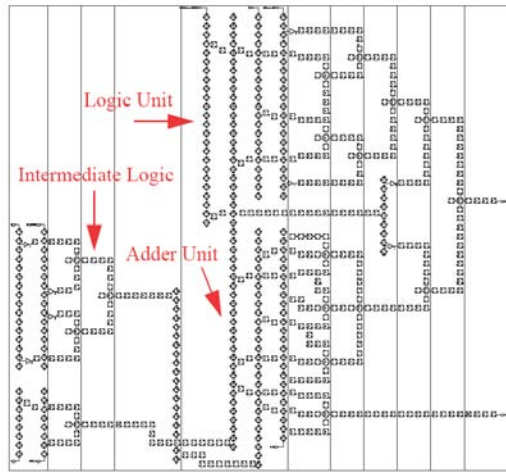


Figure 3.24 Simple ALU (From [57]. ©1999 IEEE. Reprint with permission)

3.7 COMPARISON OF NANOTECHNOLOGY DEVICES

Chapter 2 and previous sections in this chapter have provided a perspective of current-state nanoelectronic devices, which serve as potential solutions to the increasingly challenging manufacturing domain of conventional CMOS scaling. The objective of this section is to briefly depict the implementation, maturity and challenges of each nano-device. It is not the authors' intent to point out the best emerging technology as this subject is an ongoing research in scientific and engineering nano-society.

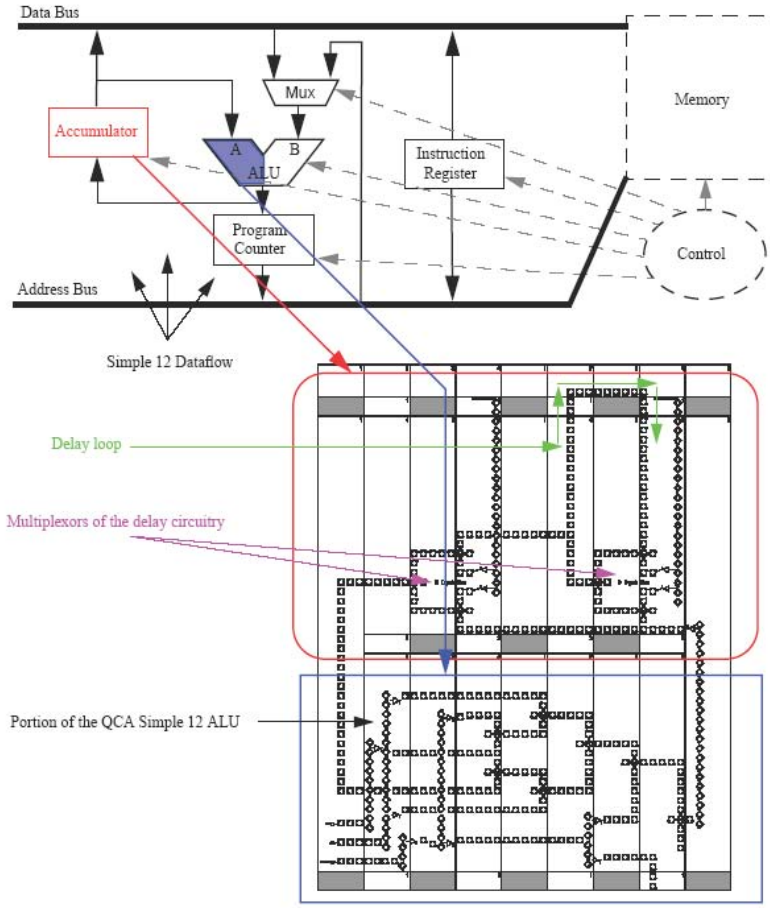


Figure 3.25 Actual QCA Representation of ALU (From [57]. ©1999 IEEE. Reprint with permission)

Table 3.1 Emerging Research Architectures

Device	Application	Status	Advantages	Disadvantages	Remarks
CNT	-Logic Elements -FET,diode -Memory	RT (room temp operation)	Ballistic transport	Difficult ctrl. over size, type, chirality and placement in circuit [59]	-Most complex circuit: ring OSC -Ballistic transport limited by CNT and bulk contact resistance [59]
QCA	-Logic Elements -Memory	-Metal: cryogenic -Molecular: RT (no circuit fabricated yet)	-High-density (10^{11} to 10^{14} devices/cm^2)	-Limited fanout[59] -Sensibility to background charge	New computation algorithms required
SET	-Logic Elements -Memory -Electrometer	RT operation	Similar design to CMOS	-Low gain -Long interconnect charge time [60] -Sensitivity to dielectric impurity [60] Sensibility to background charge	Since SET is sensitive to stray charge, SET circuits are not likely to be used for ”large CMOS type applications”; however, SET memories are more practical [61]
RTD	-RTD FET,RTT -High speed memory	RT operation	High speed	-Process Integration [59]: speed (transistor) and dynamic range (RTD) limitations	
Spin Transistor	-Spin FET -Spin value transistor: resembles a BJT	RT operation	High speed	-Inject spin polarized electrons -Ramsauer resonances [62] -Spin relaxation mechanism [62]	
Molecular devices	-Diode,FET -NEMS -Molecular QCA	RT operation	Potential to interconnect problem	Instability at high temperatures	Most complex circuit: 64-bit cross-bar array [63]

References

[1] Lent, C. S., P. D. Tougaw and W. Porod, "Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules," *PhysComp '94: Proceedings of the Workshop on Physics and Computing*, IEEE Computer Society Press, 1994, pp. 5-13.

[2] Lent, C. S., et al., "Quantum cellular automata," *Nanotechnology*,Vol. 4, No. 1, 1993, pp. 49-57.

[3] Tougaw, P. D. and C. S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 75, No. 3, 1994, pp. 1818-1825.

[4] Blick, R. H., et al., "Single-electron Tunneling Through a Double Quantum Dot: The Artificial Molecule," *Physical Review B*,Vol. 53, No. 12, 1996, pp. 7899-7902.

[5] Hofmann, F., et al., "Single Electron Switching in a Parallel Quantum Dot," *Physical Review B*, Vol. 51, No. 19, 1995, pp. 13872-13875.

[6] Waugh, F. R., et al., "Single-Electron Charging in Double and Triple Quantum Dots with Tunable Coupling," *Physical Review Letters*,Vol. 75, No. 4, 1995, pp. 705-708.

[7] Bernstein, G. H., et al., "Observation of Switching in Quantum-dot Cellular Automata Cell," *Nanotechnology*,Vol. 10, 1999, pp. 166-173.

[8] Field, M., et al., "Measurements of Coulomb blockade with a noninvasive voltage probe," *Physical Review Letters*,Vol. 70, No. 9, 1993, pp. 1311-1314.

[9] Lieberman, M., et al., "Quantum-Dot Cellular Automata at a Molecular Scale," *Annals of the New York Academy of Sciences*,Vol. 960, 2002, pp. 225-239.

[10] Timler, J. and C. S. Lent, "Power Gain and Dissipation in Quantum-dot Cellular Automata," *Journal of Applied Physics*,Vol. 91, No. 2, 2002, pp. 823-831.

[11] Amlani, I., et al., "Demonstration of a Six-dot Quantum Cellular Automata System," *Applied Physics Letters*, Vol. 72, No.17, 1998, pp. 2179-2181.

[12] Orlov, A. O., et al., "Realization of a Functional Cell for Quantum-Dot Cellular Automata," *Science*,Vol. 277, No. 5328, 1997, pp. 928-930.

[13] Lieberman, M., et al., "Quantum-dot Cellular Automata at a Molecular Scale," *Analysis of the New York Academy of Science*,Vol. 960, 2002, pp. 225-239.

[14] Cowburn, R. P. and M. E. Welland, "Room Temperature Magnetic Quantum Cellular Automata", *Science*,Vol. 287, 2000, pp. 1466-1468.

[15] Parish, M. C. B., "Modeling of Physical Constraints on Bistable Magnetic Quantum Cellular Automata", Ph.D. Thesis, University of London, UK, 2003.

[16] Amlani, I., et al., "Digital Logic Gate Using Quantum-Dot Cellular Automata," *Science*, Vol. 284, No. 5412, 1999, pp. 289-291.

[17] Korotkov A. and K. K Likharev, "Single-electron-parametron-based Logic Devices," *Journal of Applied physics*, vol. 84, no. 11, 1998, pp. 6114-6126.

[18] Toth G. and C. S. Lent, "Quasiadiabatic Switching for Metal-island Quantum-dot Cellular Automata," *Journal of Applied physics*,Vol. 85, No. 5, 1999, pp. 2977-2984.

[19] Orlov, A. O., et al., "Experimental Demonstration of a Latch in Clocked Quantum-Dot Cellular Automata," *Applied Physics Letters*,Vol. 78, No. 11, 2001, pp. 1625-1627.

[20] Walus, K., G. A. Jullien and V. S. Dimitrov, "Computer Arithmetic Structures for Quantum Cellular Automata," *Proc. Asilomar Conference,* 2003, available online: *www.qcadesigner.ca/papers/Asilomar2003.pdf*

[21] Tougaw, P. D. and C. S. Lent, "Dynamic Behavior of Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 80, 1996, pp. 4722-4736.

[22] Wang, Y. and M. Lieberman, "Thermodynamic Behavior of Molecular-Scale Quantum-Dot Cellular Automata (QCA) Wires and Logic Devices," *IEEE Transaction on Nanotechnology*,Vol. 3, No. 3, 2004, pp. 368-376.

[23] Lent, C. S., B. Isaksen and M. Lieberman, "Molecular Quantum-Dot Cellular Automata," *Journal of the American Chemical Society*,Vol. 125, No.4, 2003, pp. 1056-1063.

[24] Lu, Y. and C. S. Lent, "Theoretical Study of Molecular Quantum Dot Cellular Automata," *IEEE International Workshop on Computational Electronics*, 2004, pp. 118-119.

[25] Qi, H., et al., "Molecular Quantum Cellular Automata Cells: Electric Field Driven Switching of a Silicon Surface Bound Array of Vertically Oriented Two-Dot Molecular QCA," *Journal of the Am. Chem. Society, (JACS Articles),* Vol. 125, No. 49, 2003, pp. 15250-15259.

[26] Jiao, J., et al., "Building Blocking for the Molecular Expression of QCA, Isolation and Characterization of a Covalently Bounded Square Array of two Ferrocenium and Two Ferrocene Complexes," *Journal of the Am. Chem. Society (JACS Communications),* Vol. 125, No. 25, 2003, pp. 7522-7523.

[27] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also [Online] Available: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

[28] Hennessy, K. and C. S. Lent, "Clocking of Molecular Quantum-Dot Cellular Automata," *Journal of Vaccum Science and Technology*,Vol. 19, No. 5, 2001, pp. 1752-1755.

[29] Lent, C. S. and B. Isaksen, "Clocked Molecular Quantum-Dot Cellular Automata," *IEEE Transactions on Electron Devices*,Vol. 50, No. 9, 2003, pp. 1890-1895.

[30] Lent, C. S. and P. D. Tougaw, "A device architecture for computing with quantum dots," *Proc. of the IEEE*,Vol. 85, 1997, pp. 541-557.

[31] Frost, S. E., et al., "Carbon Nanotubes for Quantum-Dot Cellular Automata Clocking," *IEEE Conference on Nanotechnology*, 2004, pp. 171-173.

[32] Blair, E. P. and C. S. Lent, "Quantum-Dot Cellular Automata: An Architecture for Molecular Computing," *International Conference on Simulation of Semiconductor Processes and Devices*, 2003, pp. 14-18.

[33] Antonelli, D. A., et al., "Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions ," *Design Automation Conference (DAC)*, 2004, pp. 363-368.

[34] Smalley, R. E., et al., *Carbon Nanotubes: Synthesis, Structure, Properties and Applications*, Springer-Verlag, 2001.

[35] Blair, E. P., "Tools for the Design and Simulation of Clocked Molecular Quantum-dot Cellular Automata Circuits," Master's thesis, University of Notre Dame, Department of Electrical Engineering, 2003.

[36] Oskin, M., et al., "Building Quantum Wires: The Long and Short of it," *Proceedings of 30th ISCA*, No. 85, 2003, pp. 374-385.

[37] Ellenbogen, J. C. and J. C. Love "Architectures for Molecular Electronic Computers. Logic structures and an Adder Built from Molecular Electronic Diodes," *MITRE Res. Paper*, 1999.

[38] Li, M., H. Schnablegger, and S. Mann, "Coupled Synthesis and Self-assembly of Nanoparticles to Give Structures with Controlled Organization," *Nature*,Vol. 402, No. 6760, 1999, pp. 393-395.

[39] Loweth, C. J., et al., "DNA-based Assembly of Gold Nanocrystals," *Angew. Chem. Int. Ed. Engl.*,Vol. 38, No.12, 1999, pp. 1808-1812.

[40] Norris, D. J. and Y. A. Vlasov, "Chemical Approaches to Three-dimensional Semiconductor Photonic Crystals," *Adv. Mater.*,Vol. 13, No. 6, 2001, pp. 371-376.

[41] Bernstein, G. H., et al., "Electron Beam Lithography and Liftoff of Molecules and DNA Rafts," *IEEE Conference on Nanotechnology*, 2004, pp. 201-203.

[42] Fu, T. J. and N. C. Seeman, "DNA double crossover structures," *Biochemistry*, Vol. 32, 1993, pp. 3211-3220.

[43] Hu, W., et al., "High-Resolution Electron Beam Lithography and DNA Nano-Patterning for Molecular QCA," *IEEE Transactions on Nanotechnology*,Vol. 4, No. 3, 2005, pp. 312-316.

[44] Personal communication with Professor Marya Lieberman, Department of Chemistry and Biochemistry, University of Notre Dame, IN, USA.

[45] Niemier, M. T. and P. M. Kogge, "The 4-diamond circuit: A Minimally Complex Nano-scale Computational Building Block in QCA," *Proceedings. IEEE Computer Society Annual Symposium on VLSI*, 2004, pp. 3-10.

[46] Kummamuru, R. K., et al., "Power Gain in a Quantum-dot Cellular Automata Latch," *Applied Physics Letters*,Vol. 81, No.7, 2002, pp. 1332-1335.

[47] Compano, R., L. Molenkamp, and D.J. Paul, "Technology Roadmap for Nanoelectronics," *European Commission IST programme, Future and Emerging Technologies*, Available [Online]: *http://public.itrs.net/Files/2003ITRS/LinkedFiles/ERD/NanoeletronicsRdmp.pdf*

[48] Lent, C. S., M. Liu, and Y. Lu, "Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling," *Nanotechnology*,Vol. 17, No. 16, 2006, pp. 4240-4251.

[49] Timler, J. and C. S. Lent, "Maxwell's Demon and Quantum-dot Cellular Automata," *Journal of Applied Physics*, Vol. 94, No. 2, 2003, pp. 1050-1060.

[50] Niemier, M. T., M. J. Kontz, and P. M. Kogge, "A Design of and Design Tools for a Novel Quantum-dot Based Microprocessor," *Proceedings Design Automation Conference*, 2000, pp. 227-232.

[51] Ottavi, M., et al., "HDLQ: A HDL Environment for QCA Design", *ACM Journal on Emerging Technologies in Computing Systems (JETC)*,Vol. 2, No. 4, pp. 243-261, 2006.

[52] Ottavi, M., V. Vankamamidi, and F. Lombardi, "Clocking and Cell Placement for QCA", *Proc. IEEE Nanotechnology Conference*, 2006, pp. 343-346.

[53] Tang, R., F. Zhang, and Y. B. Kim, "Quantum-Dot Automata SPICE Macro Model", *ACM Great Lake Symposium on VLSI 2005*, 2005, pp. 108-111.

[54] Tang, R., F. Zhang, and Y. B. Kim, "QCA-Based Nano Circuits Design", *IEEE International Symposium on Circuits and Systems*, 2005, pp. 2527-2530.

[55] Dimitrov, V. S., G. A. Jullien and K. Walus, "Quantum-Dot Cellular Automata Carry-Look-Ahead Adder and Barrel Shifter," *IEEE Emerging Telecommunications Technologies Conference*, 2002.

[56] Frost, S. E., et al., "Memory in Motion: A Study of Storage Structures in QCA," *1st Workshop on Non-Silicon Computation*, 2002.

[57] Niemier, M. T. and P. M. Kogge, "Logic-in-Wire: Using Quantum Dots to Implement a Microprocessor," *International Conference on Electronics, Circuits, and Systems (ICECS '99)*,Vol. 3, 1999, pp. 1211-1215.

[58] Walus, K., et al., "RAM Design Using Quantum-Dot Cellular Automata," *NanoTechnology Conference*,Vol. 2, 2003, pp. 160-163.

[59] "International Technology Roadmap for Semiconductors," *Jointly Sponsored by European Semiconductor Industry Assc.,Japan Electronics and Information Technology Industry Assc., Korea Semiconductor Industry Assc., Taiwan Semiconductor Industry Assc., and Semiconductor Industry Assc.*, 2004, also Available [Online]: *http://www.itrs.net/common/2004update/2004_05_ERD.pdf*

[60] Geppert, L.,"Quantum Transistors: Toward Nanoelectronics," *IEEE Spectrum*, Vol. 37, No. 9, 2000, pp. 46-51.

[61] Raja, T., V. D. Agrawal, M. L. Bushnell, "A Tutorial on the Emerging Nanotechnology Devices," *International Conf. VLSI Design*, 2004, pp. 343-360.

[62] Pramanik, S., S. Bandyopadhyay, M. Cahay, "Why is the Spin Field Effect Transistor Elusive?" *Proc. IEEE Conference on Nanotechnology*, 2004, pp. 101-103.

[63] Chen, Y., et al., "Nanoscale Molecular-Switch Crossbar Circuits," *Nanotechnology*,Vol. 14, 2003, pp. 462-468.

[64] Javey, A., et al., "Ballistic Carbon Nanotube Field-effect Transistors," *Nature*,Vol. 424, No. 6949, 2003, pp. 654-657.

[65] Butts, M., A. DeHon and S. C. Goldstein, "Molecular Electronics: Devices, Systems and Tools for Gigagate, Gigabit Chips," *International Conference on Computer-Aided Design*, 2002, pp. 443-440.

[66] Bachtold, A., et al., "Logic Circuits with Carbon Nanotube Transistors," *Science*,Vol. 294, No. 5545, 2001, pp. 1317-1320.

[67] Avouris, P., "IBM Research: Building Carbon Nanotube Transistors," 2003.

[68] Wong, H. S. P., "Beyond Conventional Transistor," *IBM Journal of Research and Development*, Vol. 46, No. 2/3, 2003, pp. 133-168.

[69] Javey, A., et al., "Carbon Nanotube Transistor Arrays for Multistage Complementary Logic and Ring Oscillators," *Nano Letters*,Vol. 2, No. 9, 2002, pp. 929-932.

Chapter 4

QCA Combinational Logic Design

J. Huang, M. Momenzadeh, and F. Lombardi

4.1 GATE-BASED COMBINATIONAL LOGIC DESIGN

The existing literature on QCA design mostly uses a gate-based methodology [1] [2]. In a gate-based design, much like a CMOS design process, first the desired logic function of the circuit is determined and then a logic synthesis process is performed to obtain a netlist. Since in QCA the basic logic block is MV and INV, the library cell used in the logic synthesis consists of these two gates. Additionally, by fixing the polarization of one of its inputs to logic "0" ("1"), MV can be programmed into 2-input AND gate (2-input OR gate). Existing commercial logic synthesis tools for CMOS can also be used for QCA circuits by using the appropriate library cells; this is discussed in Section 4.1.1. Several MV-based logic synthesis algorithms can be found in the literature [3] [4]. The final step is to map the results of the logic synthesis to QCA layout, and assign a clocking zone to each cell. No tool is known to be able to automatically generate QCA layout given the netlist. Much of the existing QCA circuits have been designed by hand.

Inversion can be achieved in QCA using a 45 degrees cell orientation. However, it has been shown that this arrangement is not defect-tolerant [5]. Alternatively, an inverter chain (Figure 3.3) can be used to generate logic inversion. An issue associated with using the inverter chain is that rotated cells (cells rotated by 45 degrees) are employed and these cells are difficult to manufacture. Inversion can also be achieved using the INV gate (Figure 3.3). In CMOS, the INV is the simplest gate, however in QCA the INV gate is at least as large as the MV.

The gate-based design process is illustrated with an example: the design of a full adder. First the desired logic function of the full adder is:

$$Cout = AB + ACin + BCin \tag{4.1}$$

$$Sum = A\ xor\ B\ xor\ Cin \tag{4.2}$$

From logic synthesis Sum and $Cout$ can be implemented using MV and INV as follows:

$$Cout = Maj(A, B, Cin) \tag{4.3}$$

$$Sum = Maj(Cout', C, Maj(A, B, C')) \tag{4.4}$$

where $Cout'$ denotes the complement of $Cout$. It can seen that the netlist for the full adder consists of three MVs and two INVs. The resulting QCA layout is shown in Figure 4.1. From the layout it can be seen that three MVs and one INV gate are used. The other inversion is achieved using the inverter chains at the input. These inverter chains are also needed for coplanar crossing.

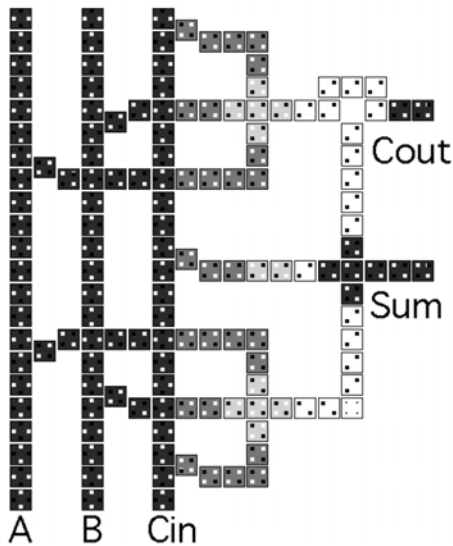


Figure 4.1 Gate-based Design of the Full Adder (From [4]. ©2004 IEEE. Reprint with permission)

4.1.1 Gate-based Design of QCA with Existing Commercial Synthesis Tools

In this section, the gate-based design of QCA is investigated using the existing commercial logic synthesis tools developed for CMOS. The gate-based QCA implementation of (combinational) logic design consists of interconnecting MVs and INVs. The MVs can be programmed into 2-input AND/OR gates by (1) using fixed polarization cells in the circuit, or (2) using global control lines. The overall structure for using global control lines is shown in Figure 4.2. There are two system-level control lines, U_0 and U_1, which are connected to the MVs. U_0 is connected to logic "0" and sets some MVs as AND gates, whereas U_1 is connected to logic "1" and sets some other MVs as OR gates. These control lines provide additional controllability because they can be regarded as extra input lines for testing purposes. This unique feature of QCA can be exploited to achieve a higher coverage and quality in the testing process [5]. However, Design-For-Testability may add a degree of complexity; for example wiring requirements are increased by using this scheme. Figure 4.3 shows a simple circuit designed either by AND/OR gates or MV-based gates.

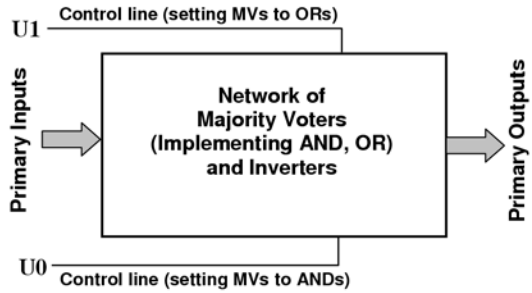


Figure 4.2 QCA Implementation of Logic Networks Using MVs (for AND and OR) and Inverters

We leverage existing commercial logic synthesis tools to synthesize logic circuits and map them into QCA library cells (such as MV and INV). Also, by setting one of the inputs of the MV to logic "1" or "0", the 2-input OR or the 2-input AND gates can be realized, respectively. The library that has been used in the analysis consists of the following cells: 2-input AND, 2-input OR, NOT, and original MV. Logic synthesis is accomplished using a commercial tool with medium mapping efforts [6]. The results for ISCAS85 and 74 series circuits are reported in Table 4.1. The effective area of a gate is used as figure of merit and is defined as the rectangular area occupied by the gate on the Cartesian plane. It is assumed that

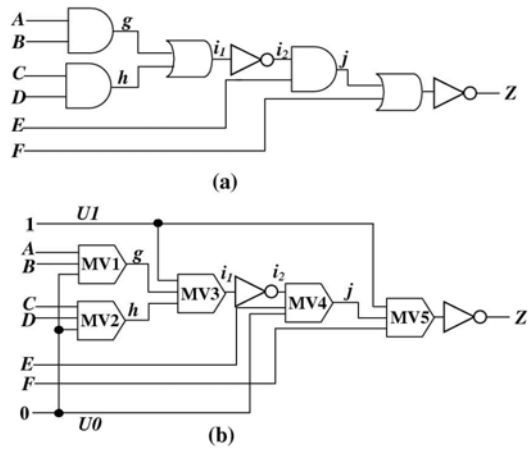


Figure 4.3 (a) AND-OR Logic Implementation (b) MV-based Implementation

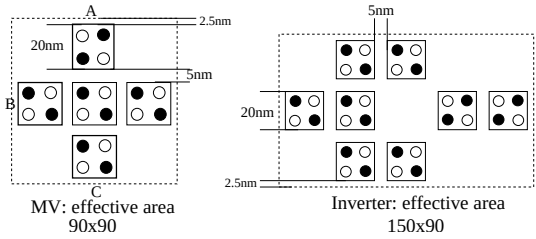


Figure 4.4 Effective Areas of MV and INV gates

$20nm \times 20nm$ cells with dot size $5nm$ are used. The cell to cell distance is set to be $5nm$ for both MV and INV, as shown in Figure 4.4. The MV consists of five QCA cells and has an effective area of $8100nm^2$, while the inverter consists of ten QCA cells and has effective area of $13500nm^2$. Let the effective area of one MV be A_{mv}, in the following the effective area will be expressed in terms of A_{mv}. For instance the effective area of INV is $1.6 \cdot A_{mv}$. The gate level implementation of an inverter is utilized in this paper (rather than the inversion generated by a 45 degree placement between two cells as part of the interconnect); this gate-level implementation has the added advantage that the outgoing wire is not offset from the incoming wire [7].

From the synthesis results, it is evident that the MV gate is not efficiently utilized by existing tools. Even for arithmetic circuits, in which there should be some perfect matches for MV (MV is the carry function of an adder), the tool does not utilize the MV gate. As AND2 and OR2 can be implemented by a single MV, then the total QCA cells for the active devices in the benchmark circuits are reported in Table 4.1.

4.2 LOGIC SYNTHESIS

Since existing commercial logic synthesis tools do not use MV efficiently, QCA requires new logic synthesis algorithms that tailors to the MV-based logic. Several MV-based logic synthesis scheme is introduced in this section.

4.2.1 AND/OR-based Logic Synthesis

The first approach, referred to as AND/OR-based synthesis, has been recently proposed specifically for QCA combinational circuits [4]. This approach reduces the number of MV gates required for computing three variable Boolean functions to facilitate the conversion of Sum-of-Product (SOP) expressions into QCA majority logic. Thirteen standard functions (exhaustively found and proposed in [4]) are utilized to completely represent all three-variable Boolean functions. Using a three cube representation, an interactive procedure is proposed [4] to generate a reduced majority gate expression that is amenable to QCA. The generated expression, however, is not always optimal, because QCA-based designs of three levels of MVs are in some cases found (it is well know that any combinational function can be implemented with any gate by a two-level circuit network).

Table 4.1
Synthesis Results for ISCAS85 and 74 Series Circuits Using MV

Circuit		AND2	OR2	INV	MV	effective area (A_{mv})
c432	behav	74	86	36	0	220
(27-ch intrpt ctrl.)	strc	74	86	36	0	220
c499	behav	158	239	134	0	620
(32-bit SEC)	strc	184	208	136	0	619
c1355	strc	235	248	141	0	718
(32-bit SEC)						
c880	strc	177	155	91	0	484
(8-bit ALU)						
c1908	strc	140	162	79	0	434
(16-bit SEC/DED)						
c2670	strc	307	245	127	2	766
(12-bit ALU/Ctrl)						
c3540	strc	352	373	129	0	940
(8-bit ALU)						
c5315	strc	855	547	277	0	1864
(9-bit ALU)						
c6288	behav	1164	1163	666	44	3481
(16×16 multiplier)	strc	985	958	481	0	2745
c7552	strc	941	744	386	0	2328
(32-bit adder/comp)						
74181	behav	40	32	20	0	105
(4-bit ALU)	strc	51	39	18	0	120
74182	behav	6	9	6	0	25
(4-bit CLA gen)	strc	11	14	6	0	35
74283	behav	26	18	14	1	68
(4-bit adder)	strc	19	8	25	0	69
74L85	behav	29	16	13	1	68
(4-bit comp)	strc	18	14	12	0	52
Average		278.38	255.43	134.9	2.29	760.94

4.2.2 Muroga's MV-based Logic Synthesis

The second synthesis approach that can be applied to QCA is the so-called MV-based synthesis [3]. This approach relies on the logic analysis of the majority function as an instance of threshold logic. Threshold logic has been extensively analyzed in the past and the majority threshold function of three variables (i.e., the MV function) is equivalent to a logic representation that can be easily implemented in QCA. Synthesis under the technique of [3] is based on identifying negated or permuted variables of a function such that restrictions can be generated to comply with the voting nature (such as agreement or disagreement) of this threshold function. An iterative process that can be extended to the general case of n variable voting functions, is required to establish the function restrictions for the specified and unspecified minterms in the SOP representation.

4.2.3 MAjority Logic Synthesizer (MALS)

The MALS is a logic synthesis tool for MV-based logic proposed in [8]. A multi-level majority network synthesis methodology is used. First the circuit is decomposed into subcircuits, each with no more than three inputs. Then the algorithm tries to find the implementation of each subcircuit using no more than four MVs. Each node is mapped into at most four MVs using the Karnaugh-map-based method. The MALS has been integrated with SIS [8]. Experimental results on MCNC benchmarks have an average reduction of 21.9% in gate count compared the logic synthesis which uses MV as a 2-input AND/OR gate.

4.3 STRUCTURAL DESIGN

Recent developments in QCA manufacturing involve molecular implementations. It is expected that homogeneous cell arrangements will be constructed by either self-assembly or large scale cell deposition on insulated substrates [9]. These manufacturing techniques are well suited to modularization. QCA design can be implemented by modularization through a simple, Manhattan-style interconnect. However, this design is expected to generate an area overhead compared to a gate-based design. This has also been encountered in CMOS: a design using a full-custom layout is usually smaller than a design using standard-cells.

In the technical literature, QCA design at the modular level has not been treated in depth. A methodology known as SQUARES has been proposed [10]. The

basic building blocks are called SQUARES, which are blocks of 5×5 cells. Logic functions (such as MV, INV) as well as interconnect (such as binary wire, fan-out, coplanar crossing) are embedded into the 5×5 grid [10]. Circuits are assembled using the SQUARES. It is assumed that each SQUARE is in its own clocking zone. Simulation of the SQUARES based circuits are performed using AQUINAS.

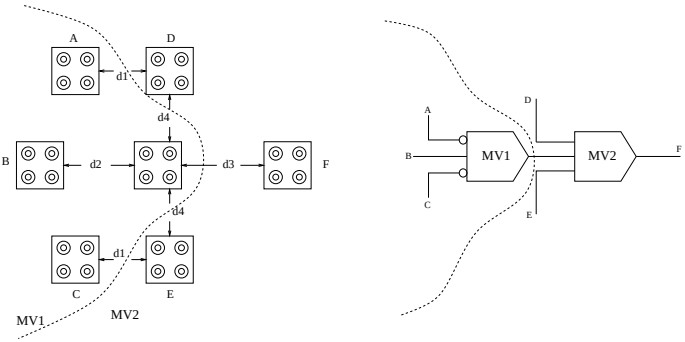
A tile-based design which utilizes 3×3 grids is introduced in Chapter 7. It will shown that the tile based design is more area efficient compared with SQUARES. Additionally, the tile based design uses the 1D clocking scheme, which is simpler and achieves much shorter delay compared with the SQUARES.

4.4 AND-OR-INVERTER (AOI) GATE

Previously in Section 4.1.1, it was shown that existing synthesis tools do not make efficient use of MV in technology mapping for synthesis of logic designs. Even for arithmetic circuits in which there should be perfect matches for the MV, the synthesis tools rarely find any matches. In this section, the design and characterization of a complex yet very small QCA logic gate: the AOI (And-Or-Inverter) gate, is proposed. The AOI gate is a universal gate with five inputs and consists of seven cells. Device characterization, testing, defect analysis and logic synthesis using the AOI gate are thoroughly investigated in this section. A detailed simulation-based characterization of the AOI gate is presented. Testing of the AOI gate is investigated at logic level and unique features of logic design based on this complex gate are identified. The AOI gate is universal: all elementary gates as well as many two-level logic functions can be implemented by a single AOI gate. Logic synthesis results with an existing commercial CAD tool show that the AOI gate is more favorable and flexible than the MV. As shown later in this section, synthesis of complex logic designs using AOI gates instead of MVs results in up to a 23.9% area reduction while the overall delay is also improved (up to a 33.4% reduction).

4.4.1 AOI Gate Characterization

Although it can be easily adapted to realize AND or OR, MV suffers from the disadvantage that it's not a universal gate and cannot offer the inverting function. Since at gate-level inversion is expensive in QCA (unlike conventional CMOS), built-in inversion is desirable. Moreover, as described previously, it has been found that the MV is not favorable in terms of technology mapping for logic synthesis. This motivates us to build a complex QCA gate with embedded AND, OR and INV functions, and with better logic synthesis capabilities.



(a) AOI Gate Layout (b) AOI Gate Schematic

Figure 4.5 The AOI Gate

Moreover, such device must exhibit stable operation such that: (1) the output must exhibit a definite polarization; (2) small misplacement of individual cells should not change the logic function of the device (i.e., the device should provide some degrees of tolerance to manufacturing process variations); (3) wiring of the device should not change its logic function.

A complex universal gate is proposed: the And-Or-Inv gate (AOI gate). The layout and corresponding logic schematic are illustrated in Figure 4.5 (the cell size is 20 nm × 20 nm and the dot size is 5 nm). This is a 7-cell gate with five input cells, one device cell and one output cell. The gate can be built from the original 5-cell MV by adding two extra inputs (cells A and C); these two inputs have an inverting effect on the center cell as from the layout of the inverter in Figure 3.3 cells in a diagonal orientation at 45 degrees exhibit an inverting function. The logic function realized by the proposed AOI gate is:

$$\begin{aligned} F &= DE + (D+E)(A'C' + A'B + BC') \\ &= Maj(D, E, Maj(A', B, C')) \end{aligned} \tag{4.5}$$

where $Maj()$ is the 3-input majority function, A' denotes the logic inversion of A. By simulation it has been found that this AOI gate is relatively stable, i.e. a marginal cell misplacement does not change its logic function. It has been found that the placement in Table 4.2 (see Figure 4.5) yields an AOI gate performing the function described above. A semisymmetric and stable configuration with $d1$ = $d3$ = $d4$ = $25nm$, $d2$ = $35nm$, has been used here.

Table 4.2
Cell Placement in the AOI (Not Wired) Gate

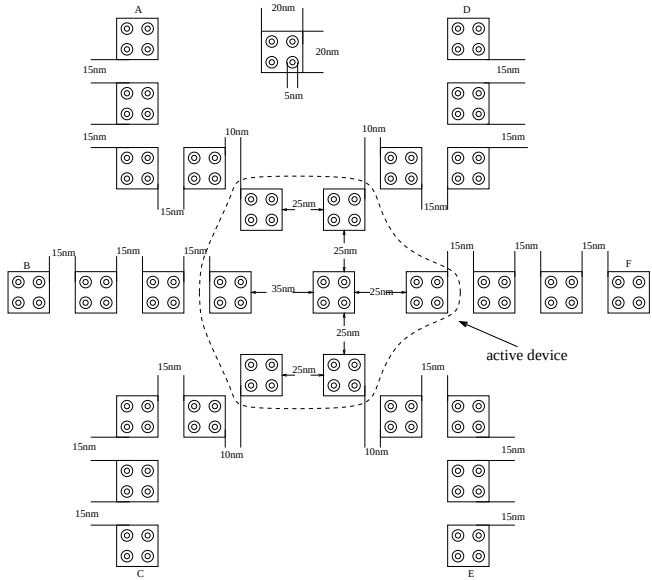
$d1$ nm	$d2$ nm	$d3$ nm	$d4$ nm
20	30-40	20-40	20
25	30-40	20-40	20
30	35-40	20-40	20
25	35-40	25-40	25
30	35-40	25-40	25

The AOI gate is logically equivalent to a concatenation of two MVs with two complemented inputs (A and C). The layout of the AOI gate consists of two nested MVs: MV1 and MV2; these MVs are separated by a dotted line in Figure 4.5(a). $MV1$ performs the function $MV1 = Maj(A', B, C')$. It has been shown in our previous work [5] that the horizontal input (i.e., B) has the strongest influence on the center cell in an MV. Therefore in the AOI gate, cell B is placed farther away than A and C (see Table 4.2). Since A and C tend to have an inverted effect on the center cell, $MV1$ is the majority of A', B and C'. The second MV is $MV2 = Maj(D, E, MV1)$.

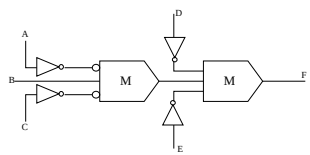
The proposed wiring scheme for the AOI gate is shown in Figure 4.6; the active AOI gate has $d1 = d3 = d4 = 25nm$, $d2 = 35nm$. As our previous research in [5] indicates, when two binary wires are placed sufficiently close, they may interfere with each other, similar to crosstalk in conventional CMOS circuits. The main challenge in wiring the AOI gate is the separation of the input/output binary wires such that they do not interfere (while still preserving the original logic function). By simulation it has been found that the wire for cell A and D (also C and E) must have a distance of more than $25nm$, and wiring for inputs A, C, D and E has an inverting effect. This is due to the 45 degrees orientation between the wires (of input A, C, D, E) and active device. As a result, at logic level wiring the AOI gate adds additional inverters at inputs A, C, D and E. The device wiring layout and its schematic are shown in Figure 4.6. Based on simulation, the AOI gate with the above wiring scheme has been found to be reasonably stable.

4.4.2 Defect Characterization of the AOI Gate

In this section, the robustness of the AOI gate is investigated. The motivation for conducting this study is to make sure the proposed design is robust with respect to



(a) Wired AOI Gate



(b) Schematic of Wired AOI Gate

Figure 4.6 The Wired AOI Gate

manufacturing process variations. The basic functionality of a QCA device is based on the Coulombic interactions among neighboring QCA cells, which depend on the accuracy and geometry of their implementation. Various configurations of the AOI gate have been studied using the QCADesigner [11] v1.20 simulation tool. The bistable simulation engine has been used.

Cell misplacement defects are considered here. A *cell misplacement* is a defect in which the defective cell is misplaced from its intended position. In this work, it is assumed that each individual cell functions correctly and cell misplacement are simulated with respect to the central cell under different distance conditions. The investigation of the behavior of the AOI gate in the presence of cell misplacements establishes not only its defect tolerance, but it also gives an insight into cell interactions within the AOI gate.

In the simulation the fault free AOI gate (as in Figure 4.5) has $d1 = d3 = d4 = 25nm$, $d2 = 35nm$. The input and output cells of the AOI gate are then moved with respect to the central cell and record the logic function performed by the AOI gate. Some of the simulation results with $d1 = d4 = 25nm$ are reported in Table 4.3. Similar defective patterns occur with $d1 = d4 = 20nm$ or $d1 = 25nm, d4 = 20nm$ or $d1 = 30nm, d4 = 20nm$ or $d1 = 30nm, d4 = 25nm$.

An important result observed from Table 4.3 is that the horizontal input (cell B) has greater influence on the central device cell than the other inputs, which confirms our results in [5]. If Cell B is placed sufficiently close to the central cell, the output follows Cell B, and the whole AOI gate acts as a binary wire with input B. Two other interesting patterns can be observed in which the AOI gate behaves as an MV with $F = DE + BE + BD = Maj(B, D, E)$ or an MV with inversions at some inputs with $F = D'E' + BD' + BE' = Maj(B, D', E')$. In these two cases, B is closer to the central cell than in the fault free case and cancels the effect of A and C. When the output F is placed sufficiently far from the central cell, no polarization can be observed at the output (indicated by $F = Z$ in the table). Also when B is placed far away from the central cell, some input combinations cause the output to show no polarization at all. These results are consistent because in QCA, information is transmitted via Coulomb interactions, so the larger the distance between two cells, the weaker the interactions are. It can also be concluded that the AOI gate is reasonably robust as a small misplacement does not change the functionality.

Table 4.3
Defect Characterization of the AOI Gate

d1	*d4*	*d2*	*d3*	Output Function F
25	25	5-10 15 20 25	5-40 15-40 20-40 25	B
		15 20 25-30	5-10 5-15 5-20	$D'E' + BD' + BE' = Maj(B, E', D')$
		25 30	30-40 25-40	$DE + BE + BD = Maj(B, D, E)$
		35-40	5-20	$D'E' + (D' + E')(A'B + BC' + A'B'C')$
		35-40	25-40	Normal Operation
		≥45		For some input combinations F=Z (no polarization)
			≥45	For all input combination F=Z (no polarization)

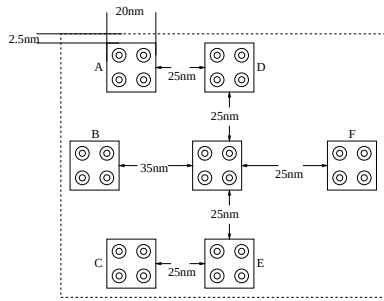


Figure 4.7 Effective Area of the AOI Gate

4.4.3 Logic Synthesis Using the AOI Gate

In this section, logic synthesis results of AOI gate are compared with those of the MV based synthesis results. It is assumed that $20nm \times 20nm$ cells with dot size $5nm$ are used. The cell to cell distance is set to be $5nm$. As explained in Section 4.1.1, the MV has effective area $90nm \times 90nm = 8100nm^2$ (see Figure 4.4). Let the effective area of one MV be A_{mv}, the AOI gate has an effective area of 1.771 A_{mv}, as shown in Figure 4.7. If the same logic is implemented with MV and inverters, an effective area of 6 A_{mv} is needed. Further, it will be shown next that implementing basic logic functions with the AOI gate instead of MV and INV in most cases substantially reduces area overhead. Thirteen standard functions are introduced in [4] to represent all 256 three-variable Boolean functions. Let the three Boolean variables be a, b and c. Then the thirteen standard functions is shown in Table 4.4, where A, B and C can be mapped to any one of a, b, c, a', b', c'. For example $F = a'b + bc'$ and $F = bc + ca$ can both be represented by the same standard function $F = A'B + BC'$. The simplified MV/INV implementation of the thirteen standard functions proposed in [4] is illustrated in Figure 4.8. These standard functions can also be realized using the AOI gate, which is also illustrated in Figure 4.8. Note that when utilizing the built-in inversion of the AOI gate, no extra inverter is needed. Table 4.4 shows the comparison in terms of gate count as well as total effective area of active gates between the two implementations. Clearly, except for three very simple logic functions that can be implemented with a single MV ($F = A$, $F = AB$ and $F = AB + BC + AC = Maj(A, B, C)$), the AOI-based design achieves up to 60.6% area savings. In practice MV can be used to construct the three simple functions while the AOI gate is used to implement the rest of the more complex logic functions.

However, the implementation of the thirteen standard functions presented in [4] is not fully minimized. It has been proved in [3] that any three-variable function can be implemented with two levels of MVs (at most four MVs). Using the Karnaugh-Map method proposed in [3], the thirteen functions can be implemented as shown in Figure 4.9. Compared to the implementation in [4], functions 4 and 11 are simplified: instead of using 5 MVs, only 4 MVs are needed. Obviously, even if the two level MV implementation is used for the thirteen standard function, the AOI implementation still has a significant area advantage.

Logic synthesis of benchmark circuits using the AOI gate have also been investigated. By using some of the inputs as programming inputs and setting them to logic "0" or "1", the AOI gate can be programmed to realize a variety of two-level logic functions. Figure 4.10 shows various logic functions implemented by the AOI

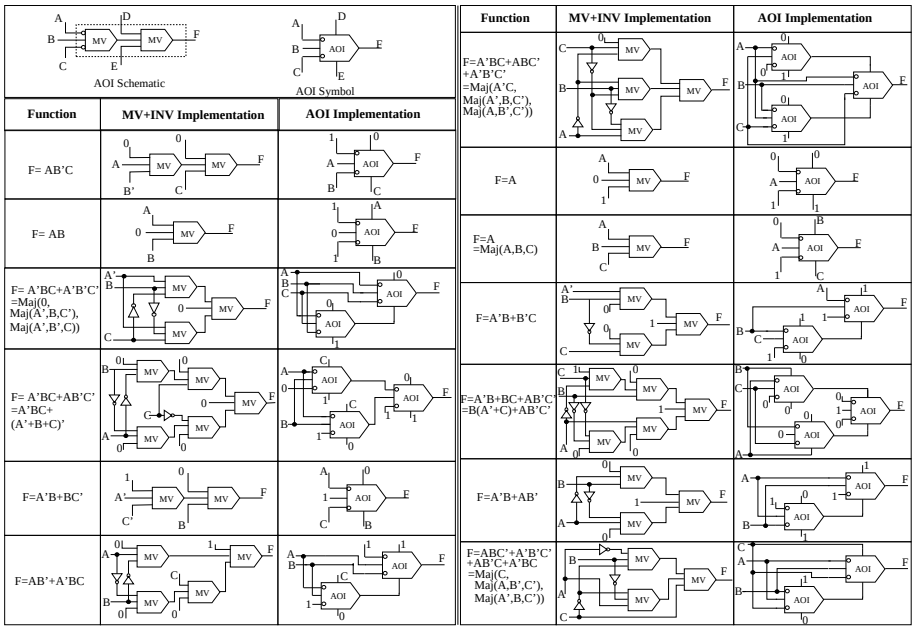
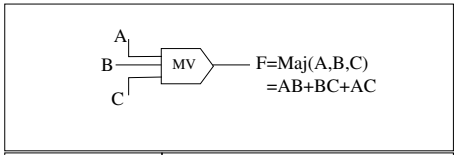


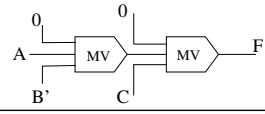
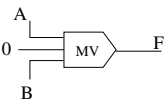
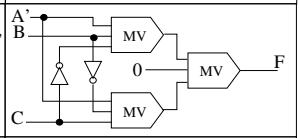
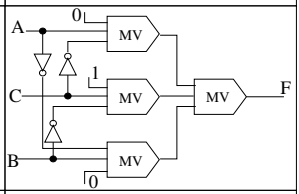
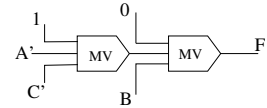
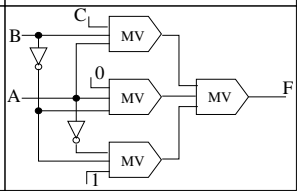
Figure 4.8 MV+INV and AOI Implementation of Thirteen Standard Functions

Functions		# MV & INV			AOI		
		# of MV	# of INV	eff. area (A_{mv})	# of AOI	eff. area (A_{mv})	improve-ment
1	F=AB'C	2	0	2	1	1.771	11.45%
2	F=AB	1	0	1	1	1.771	-77.10%
3	F=A'BC+A'B'C'	3	2	6.334	2	3.542	44.07%
4	F=A'BC+AB'C'	5	3	10	3	5.313	46.87%
5	F=A'B+BC'	2	0	2	1	1.771	11.45%
6	F=AB'+A'BC	4	2	7.334	2	3.542	51.70%
7	F=A'BC+ABC' +A'B'C'	4	3	9.001	3	5.313	40.97%
8	F=A	1	0	1	1	1.771	-77.10%
9	F=AB+AC+BC	1	0	1	1	1.771	-77.10%
10	F=A'B+B'C	3	1	4.667	2	3.542	24.10%
11	F=A'B+BC+AB'C'	5	3	10	3	5.313	46.87%
12	F=AB+A'B'	3	2	6.334	2	3.542	44.07%
13	F=ABC'+A'B'C' +AB'C+A'BC	3	3	8.001	2	3.542	55.73%
Average		2.85	1.46	5.28	1.85	3.27	11.23%

Table 4.4

MV+INV vs AOI Expression of Thirteen Standard Functions



Function	2–level MV Implementation
F= AB'C	
F= AB	
F= A'BC+A'B'C' =Maj(0, Maj(A',B,C'), Maj(A',B',C))	
F= A'BC+AB'C' =Maj(AC', B'+C,A'B)	
F=A'B+BC'	
F=AB'+A'BC =Maj(AB',A'+B', Maj(A,B,C))	

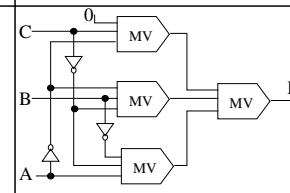
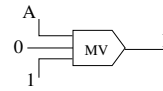
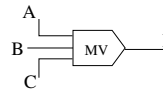
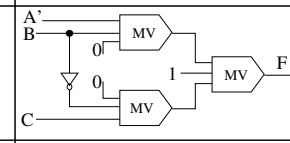
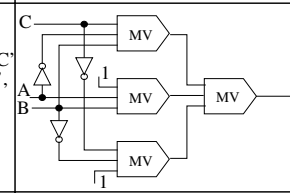
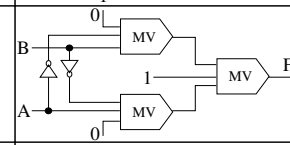
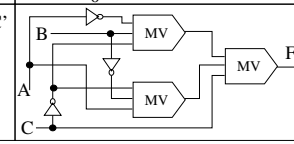
Function	2–level MV Implementation
F=A'BC+ABC' +A'B'C' =Maj(A'C, Maj(A',B,C'), Maj(A,B',C'))	
F=A	
F=Maj(A,B,C)	
F=A'B+B'C	
F=A'B+BC+AB'C' =Maj(A+B,B'+C', Maj(A',B,C))	
F=A'B+AB'	
F=ABC'+A'B'C' +AB'C+A'BC =Maj(C, Maj(A,B',C'), Maj(A',B,C'))	

Figure 4.9 Two-Level MV Implementation of Thirteen Standard Functions

gate. For example, if $A = D = 1$, then an ANDOR gate performing $F = BC' + E$ is obtained. Or if $B = D = 0$, $E = 1$, then a 2-input NAND gate with $F = (AC)'$ is obtained. Therefore, the AOI gate is universal and any combinational circuit can be implemented using only AOI gates.

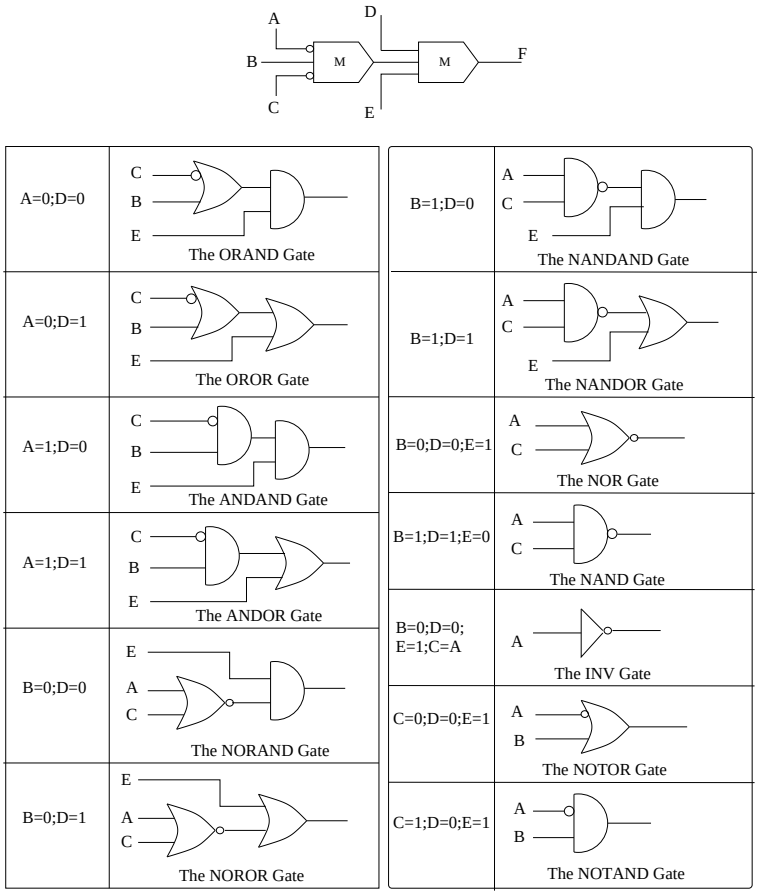


Figure 4.10 Various Gates Constructed by AOI Gate

Similar to the MV, existing commercial synthesis tools cannot find a perfect match for the AOI gate. However, the elementary gates (as well as some gates that performs a two-level logic function) constructed by the AOI gate (by setting

some of the inputs of the AOI gate to logic "1" or "0") can be efficiently used by existing commercial synthesis tools [6]. Moreover for these functions and the different QCA implementations, two cases can be distinguished. (1) If for the AOI or MV implementation, both the true and complemented values of an input signal are required (say A and A') an inverter is added to both implementations. (2) If only A or only A' is needed, then no inverter is added to the AOI implementation because inversion can be internally generated.

The same logic synthesis software and settings as used in Section 4.1.1 are used here to obtain synthesis results using the AOI gate. The library used contains 13 cells, derived from only one AOI gate, inclusive of 8 two-level gates (*not-or-and*, *not-or-or*, *not-and-and*, *not-and-or*, *nor-and*, *nor-or*, *nand-and*, and *nand-or*) and 5 one-level gates (*nor*, *nand*, *not*, *not-or* and *not-and*), as shown in Figure 4.10. The effective area for the AOI gate is $14375nm^2$. The synthesis results for the ISCAS85 benchmarks and some of the 74 series circuits are shown in Table 4.5. Columns 3 and 4 show the results for logic synthesis using MV and INV. Column 3 is the number of QCA gates used, while column 4 is the effective area. The results using the AOI gate are presented in columns 5 to 7. Column 5 shows the number of one-level and two-level gates used. The total effective area using AOI is in column 6. Improvements against MV-based results in terms of effective area are shown in column 7.

The synthesis results show that the tool effectively utilizes all one-level and two-level logic functions from the AOI gate. In all but one case, the AOI-based implementation results in an area optimization of up to 23.9% compared to an MV-based implementation. Moreover, the number of AOI gates used in the critical paths is smaller than for the MV and inverter gates because a single AOI gate can implement many two-level logic functions. Our synthesis results show that the number of gates in the critical path is up to 33.4% less when using AOI instead of MV and inverter. Also, the delay of an AOI gate is almost the same as for an MV gate. Hence, the overall delay of each of these benchmark circuits is also reduced.

4.4.4 Conclusion

In this section, the design and characterization of a novel, complex yet efficient QCA logic gate called the AOI gate has been proposed. A detailed simulation-based analysis and a characterization of QCA defects have been presented. Simulation results have shown that the presented AOI gate is robust to manufacturing process variations. The AOI forms a universal logic gate: all elementary gates can be implemented by using the AOI gate. Moreover, many two-level logic functions can

Table 4.5
Synthesis Results of ISCAS85 and 74 Series Circuits for Various Gates Constructed with AOI

Circuit		# MV & INV		Various AOI		
		gates	effective area(A_{mv})	1-lev.+ 2-lev. gates	effective area(A_{mv})	improve-ment
c432	behav	196	220	14+83	172.15	21.75%
	strc	196	220	17+102	211.19	4.01%
c499	behav	531	620	64+247	551.93	11.03%
	strc	528	619	16+265	498.69	19.39%
c1355	strc	624	718	20+292	553.70	22.88%
c880	strc	420	484	54+197	445.45	7.9%
c1908	strc	381	434	15+187	358.49	17.34%
c2670	strc	681	766	93+326	743.60	2.88%
c3540	strc	854	940	82+427	903.32	3.9%
c5315	strc	1679	1864	227+814	1847.45	0.87%
c6288	behav	3037	3481	225+1618	3270.76	6.04%
	strc	2424	2745	263+1601	3308.02	-20.53%
c7552	strc	2071	2328	204+1009	2152.70	7.54%
74181	behav	92	105	14+39	94.06	10.7%
	strc	108	120	18+42	106.48	11.27%
74182	behav	21	25	4+7	19.52	21.91%
	strc	31	35	10+8	31.94	8.73%
74283	behav	287	68	12+29	72.76	-6.48%
	strc	245	69	10+25	62.11	9.54%
7485	behav	59	68	0+29	51.47	23.94%
	strc	44	52	0+23	40.82	21.5%
Average		671	760.94	64.86+350.95	737.93	9.82%

be directly implemented by a single AOI gate. Unlike a conventional MV, the AOI gate operates quite favorably in terms of digital logic synthesis. This gate can be efficiently used by existing synthesis tools. As shown by simulation, synthesis of complex designs using the AOI gates (instead of MVs) results in up to a 23.9% area reduction while the overall delay is also improved (up to a 33.4% reduction).

References

[1] Niemier, M.T. and P.M. Kogge, "Problems in designing with QCAs: layout=timing," *International Journal of Circuit Theory and Applications*,Vol. 29, No. 1, 2001, pp. 49-62.

[2] Walus, K., et al., "RAM Design Using Quantum-Dot Cellular Automata," *NanoTechnology Conference*,Vol. 2, 2003, pp. 160-163.

[3] Muroga, S., *Threshold Logic and Its Applications*, New York, NY: John Wiley and Sons Inc., 1971.

[4] Zhang, R., et al., "A Method of Majority Logic Reduction for Quantum Cellular Automata," *IEEE Trnsactions on Nanotechnology,* vol 3, No. 4, 2004, pp. 443-450.

[5] Tahoori, M. B., M.Momenzadeh, J. Huang, F. Lombardi, "Defects and Faults in Quantum-Dot Cellular Automata", VLSI Test Symposium (VTS), 2004, pp. 291-296.

[6] "Design Compiler Technology Backgrounder", also available online: *http://www.synopsys.com/products/logic/design_comp_tb.pdf*, 2002.

[7] Tougaw, P. D. and C. S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 75, No. 3, 1994, pp. 1818-1825.

[8] Zhang, R., P. Gupta, N. K. Jha, "Synthesis of Majority and Minority Networks and Its Application to QCA, TPL, and SET Based Nanotechnologies", *IEEE Conference on VLSI Design held jointly with International Conference on Embedded Systems Design*,, 2005.

[9] Bernstein, G. H., et al., "Electron Beam Lithography and Liftoff of Molecules and DNA Rafts," *IEEE conference on Nanotechnology*, 2004, pp. 201-203.

[10] Berzon, D. and T. J. Fountain, "A Memory Design in QCAs Using the SQUARES Formalism," *Proceedings Ninth Great Lakes Symposium on VLSI*, 1999, pp. 166-169.

[11] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also available online: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

Chapter 5

Logic-Level Testing and Defect Characterization

M. Momenzadeh, J. Huang, and F. Lombardi

This chapter investigates logic-level testing and defect characterization aspects of QCA circuits. In the first part of this chapter, logic-level testing for MV-based as well as AOI-based QCA circuits has been analyzed. Unique test properties of QCA circuits have been identified. C-testability (constant-testability) of a 1-dimensional array of MVs is discussed. The second part of this chapter deals with the robustness of the QCA and QCA circuits. Defect characterization has been pursued in detail.

5.1 LOGIC-LEVEL TESTING

In logic-level testing, a set of vectors are applied to the primary inputs of the circuit under testing. The primary outputs of the circuit are then collected and analyzed. A fault is said to be detected if, for at least one test vector, at least one of the outputs is different from the expected value. Since there are too many manufacturing defect mechanisms to be targeted, testing is done based on *fault models*, which are abstractions of defects at the logic level. The properties of an appropriate fault model can be described as: (1) the test sets generated using this fault model can detect a high percentage of realistic defects and (2) test-generation complexity is not excessive. Moreover, the fault model should capture the behavior of the majority of defects at the logic level. Although for CMOS only a small percentage of actual defects behaves like stuck-at faults, the stuck-at fault model is still widely used

because test sets generated based on this model have high coverage. So, despite the fact that most defects for a deep sub-micron CMOS process do not behave according to a stuck-at fault at physical level, test sets that are generated based on the stuck-at fault model still detect a large percentage of realistic defects. Furthermore, test generation using the stuck-at fault model is not complex. Other fault models may describe the nature of defects more precisely, however the generation of test sets using these models is so complex that it is impractical for large circuits. Therefore, it is important to investigate the effectiveness of the stuck-at fault model for QCA defects even though the defect mechanisms in QCA cannot be modeled as stuck-at faults at the physical level.

5.1.1 Stuck-at Test Properties of MV-based Circuits

The overall structure of a QCA implementation for (combinational) logic designs is shown in Figure 5.1. The block consists of an interconnection of MVs and INVs. There are two system-level control lines, U_0 and U_1, which are connected to MVs. U_0 is connected to logic "0" and sets some majority voters to the AND function, whereas U_1 is connected to logic "1" and sets the other MVs to the OR function. A simple example is shown in Figure 5.2. These control lines provide additional controllability because these lines can be seen as extra input lines during testing time. This unique feature of QCA can be exploited to achieve a higher test coverage and quality. However, no Design-For-Testability scheme comes for free; for example wiring requirements are increased by adding the global control lines generating additional wire crossings in the design. There has been some research in QCA placement and routing problems [1].

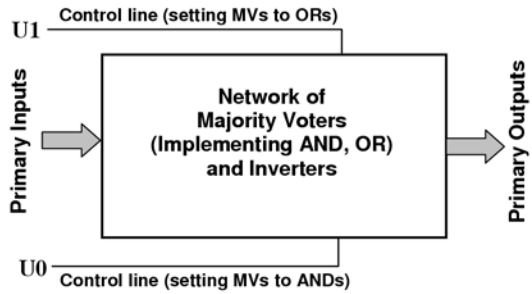


Figure 5.1 QCA Implementation of Logic Networks Using MVs (Implementing AND and OR) and Inverters

Since logic designs are implemented as a network of MVs and INVs (as the universal logic set) in QCA technology, it is important to investigate the properties of these networks, especially for test execution. As shown through the following statements, these networks have unique and interesting testing features which cannot be achieved in conventional CMOS implementations.

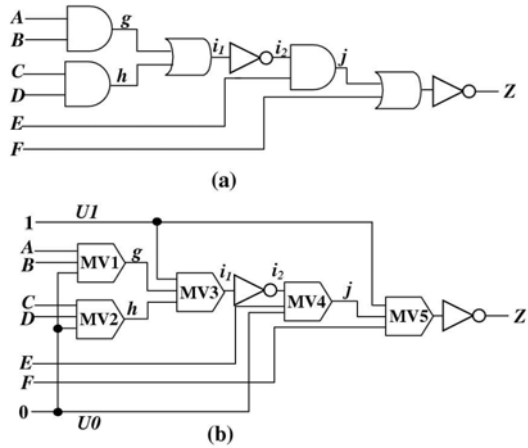


Figure 5.2 (a) A Simple AND-OR Logic (b) MV-based Implementation

Consider a majority voter with input lines A, B, and C, and output line Z (where $Z = AB + AC + BC$).

Property 1. *Consider a majority voter with input values a, b, and c, (for lines A, B, and C, respectively) and output z. If all inputs are flipped,* $abc \rightarrow a'b'c'$*, then the output will be also flipped,* $z \rightarrow z'$*. (where* A' *is the complement of* A*)*

Note that this is not the case for other logic functions such as AND, NOR, and so on. For example, consider a three input AND gate with inputs 100 and output 0. If the inputs are flipped to 011, then the output will remain 0.

Property 2. *If there is inversion at any input and/or the output of the majority voter, property 1 still holds.*

Property 3. *Consider a majority voter with input pattern* abc *(for lines A, B, and C, respectively). The stuck-at-v fault on any input or output line of the voter is detectable (the fault effect appears at the output line) by* abc *if and only if the stuck-at-*v' *fault on that line is detectable by* $a'b'c'$*.*

Proof. Consider l stuck-at-v fault. If l is an input line, consider l to be A, without loss of generality. The fault is detected if and only if the value of a is v'

and the other inputs (b and c), have opposite values. As a result, a' is v and b' and c' have opposite values. Hence, $a'b'c'$ detects the stuck-at-v' fault for l.

Again, this property does not hold for other logic functions. As an example, consider a two-input AND gate with test vector 11 that detects stuck-at-0 at both the top input and the bottom input. The complement of this vector, 00, does not detect any single stuck-at-1 on the inputs.

Property 4. *If there are some inversions at any inputs and/or the output of the majority voter, then property 3 still holds.*

The interesting property of majority voters is that the above properties hold for any arbitrary network of majority voters and inverters.

Property 5. *Consider an arbitrary network of majority voters and inverters with primary input vector V. If all bits of V are flipped, $V \rightarrow V'$, all nodes in the network will be flipped.*

Proof. The proof is based on induction on the level (distance) of each majority voter in the network from the primary inputs, by forming a topological order of the majority voters in the network. The step of induction is property 2.

Property 6. *Consider an arbitrary network of majority voters and inverters with primary input vector V. For any node n in the network, n stuck-at-u is detected by V, if and only if n stuck-at-u' is detected by V'.*

Proof. The proof is similar to the proof of property 5. The step of induction is property 4.

Properties 5 and 6 are very interesting and proved unique features of a network of majority voters and inverters. Based on property 5, the test vector pair (V, V'), where V is any arbitrary vector, causes a transition on all nodes of the network. Also, the three vectors (V, V', V) cause both fall and rise transitions on all nodes in the network. Hence, a 100% toggle fault coverage is applicable for this test set.

Based on property 6, the fault list for any network of majority voters and inverters can be divided into two parts: just one fault per each node, because if a vector V detects one stuck-at fault on that node, V' will detect the other stuck-at fault on that node. As a corollary, this feature can be exploited to reduce the size of the fault list, and hence Automatic Test Pattern Generation (ATPG) execution, for the control inputs (to be generated by ATPG) into half.

To generate tests for detecting stuck-at faults in a network of MVs and INVs, conventional (combinational) ATPG tools can be exploited. The network of MVs and INVs is first transformed into a hierarchical gate-level netlist. Each MV is replaced by a hierarchical cell implementing the majority function. We only consider pin faults on the inputs of these hierarchical cells that correspond to the

inputs of MVs. As explained above, only half of the pin faults must be considered for test generation.

5.1.2 Test Set for MVs

Consider the simple AND-OR structure shown in Figure 5.3(a) and a possible implementation using MVs in Figure 5.3(b). Note that there is no built-in VDD or ground lines in quantum dot based designs. There are two extra inputs connected to logic "1" and logic "0" to connect some selected inputs of MV to implement AND and OR logic functions. We refer to these inputs as the control lines. The input line of MV, which is connected to a control line, is called control input (the control line is a fanout stem and the control inputs of MVs are fanout branches connected to the control line). The other inputs are called non-control inputs.

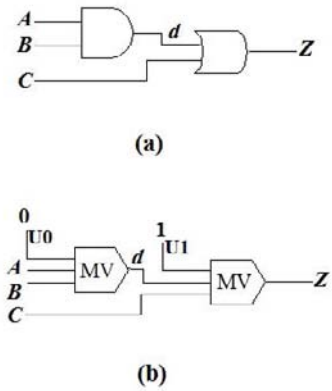


Figure 5.3 (a) An AND-OR Circuit (b) Implementation by MVs

The exhaustive testing of the circuit in Figure 5.3(a) needs all eight combinations of the three inputs. The minimum test set with 100% single stuck-at fault coverage for this circuit contains four vectors. These vectors are $ABC = (010, 100, 101, 110)$, The fault list is $A/1$ (A stuck-at 1), $A/0$, $B/1$, $B/0$, $C/1$, $C/0$, $d/1$, $d/0$, $Z/1$ and $Z/0$. However, 100% stuck-at coverage for the same fault list contains only two vectors for the implementation using MVs, shown in Figure 5.3(b). These vectors are $(ABCU_0U_1) = (11100, 00011)$. In the first test vector, both control inputs, U_0 and U_1, are connected to 0. This vector detects $A/0$, $B/0$, $C/0$, $d/0$, and $Z/0$. The second input connects all control inputs to 1 and sets the

primary inputs, A, B, and C, to 0. Therefore, the MVs implement OR functions. This vector detects all stuck-at-1 faults, namely, $A/1$, $B/1$, $C/1$, $d/1$, and $Z/1$. This reduced test set is achievable due to the specific features of the MV network and extra controllability offered by the control inputs.

Note that any 100% stuck-at coverage test set for the original circuit of Figure 5.3(a) detects no stuck-at faults on the control lines of MVs, namely $U_0/1$, $U_0/0$, $U_1/1$, $U_1/0$. This includes all test sets generated prior to mapping the design into MVs, and the above pair of vectors. Testing of a control line of MV for stuck-at faults requires that the two other inputs of MV have opposite values. By applying 1 and 0 on the control line, stuck-at-0 and stuck-at-1 faults on the control line will be detected, respectively. If for a particular vector at the primary inputs, the two non-control inputs of each MV have different values, then all stuck-at faults on the control lines can be detected by only two test vectors.

In the above example, the following two vectors must be added to the test set to detect control line faults: $(ABCU_0U_1) = (10011, 01100)$. The first (second) vector detects stuck-at-0 (stuck-at-1) faults on the control lines. Note that the non-controlling inputs of each MV have opposite values in each test vector.

Now consider a more complex example as shown in Figure 5.4(a), with a possible implementation by QCA MVs in Figure 5.4(b). This network requires at least seven test vectors for 100% single stuck-at fault coverage. However, the circuit in Figure 5.4(b) requires only two vectors to achieve 100% fault coverage for the same fault list (all stuck-at faults on nodes $A, B, C, D, E, F, g, h, i, j, Z$). These two vectors are: $(ABCDEFU_0U_1) = (00000011, 11111100)$. In this case, testing for stuck-at faults on control lines cannot be accomplished by two test vectors as in the previous example. It is not possible to simultaneously set the non-control inputs of MV1, MV2 and MV3 to opposite values (i.e., AB, CD, and gh) because the control inputs of MV1 and MV2 are connected to the same control line. This results in more than two test vectors for detecting stuck-at faults on all control inputs and control lines. Generating test sets for a network of MVs and design-for-testability of QCA circuits have been presented in [2].

5.1.3 C-Testability of MV-based Designs

In this section, C-testability (constant-testability) of a 1-dimensional array of MVs is discussed. We present a 100% stuck-at fault test set for a chain of n MVs, as shown in Figure 5.5. A 100% single stuck-at fault test set for a single MV has a minimal length of 4, such as $\{$010,011,100,101$\}$, $\{$001,011, 100,101$\}$, $\{$001,010,101,110$\}$.

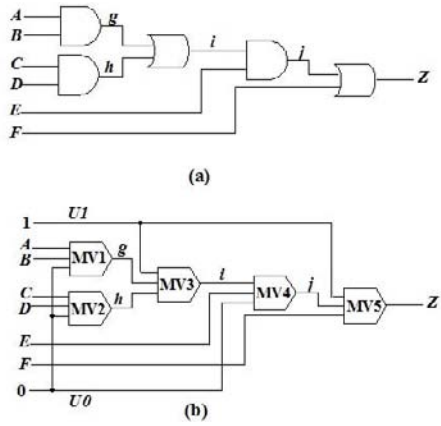


Figure 5.4 (a) Network of AND-OR (b) Implementation by MVs

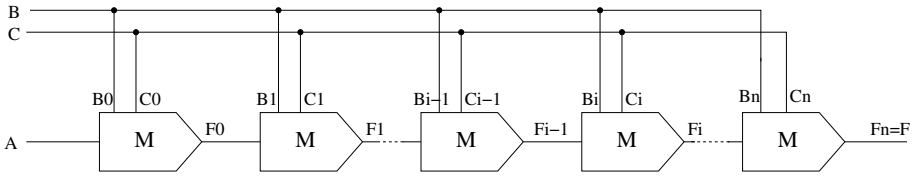


Figure 5.5 A Chain of n MVs

Table 5.1
Detecting SSF in MV Chain

SSF	test vectors (A B C)			
	001	010	101	110
A/0			x	x
A/1	x	x		
Bi/0				x
Bi/1	x			
Ci/0			x	
Ci/1		x		
Fi/0			x	x
Fi/1	x	x		

As shown in Figure 5.5, n MVs are concatenated into a 1-D (one-dimensional or linear) chain. A is the primary data input, and B_i,C_i are the control inputs. By applying $B_iC_i = 01$ for all MVs in the chain and setting the primary input A to 0, any stuck-at-1 fault on F_i and B_i will be detected. This occurs because the MV chain is effectively converted to a chain of OR gates with inputs F_i and B_i. As the test vector for these inputs is 00, any stuck-at-1 fault will be detected. Similarly, by applying $A = 1$, any stuck-at-0 fault on F_i or C_i will be detected (a 1-D chain of AND gates with inputs F_i and C_i). To detect B_i stuck-at-0 or C_i stuck-at-1 faults, we need more vectors: $B_iC_i = 01$ and setting A to 1 and 0, respectively.

A/1 (A stuck-at-1) can be detected by vectors 001 or 010. A is set to 0 to sensitize the fault, $BC = 01$ or 10 will propagate the fault to F. Similarly, **A/0** is detected by 101 and 110. **Bi/1** is detected by 001. In this case the faulty gate is the MV i with inputs F_{i-1}, B_i,C_i and output F_i. B is set to 0 to sensitize the fault. As $BC = 01$, then A will be propagated to F_{i-1} (i.e., F_{i-1} = A = 0). So the stuck-at fault at B_i can be propagates to F_i because the other inputs of the gate are 0 and 1. Also $BC = 01$, so the faulty value will be propagated to the primary output F. Similarly, **Bi/0** is detected by 110. **Fi/1** is detected by vectors 001 or 010. Since $BC = 01, 10$, $A = 0$ if it is propagated to F_i to sensitize the fault. Then, the faulty value will propagate down the chain and reach the primary output F. Similarly, **Fi/0** is detected by 101 and 110. The single stuck-at fault (SSF) detected by each test vector is shown in Table 5.1.

Hence, any 1-D chain of MVs independent of its length n can be tested for all stuck-at faults by only 4 vectors, i.e., it is C-testable. This can be generalized to

a two-dimensional (2-D) network of MVs. Note that in the 1-D chain any number of MVs can be faulty; detection with 100% coverage of multiple faulty MVs is possible due to the AND-OR nature of the MV chains during testing by getting the values of the control inputs.

5.2 DEFECT CHARACTERIZATION OF DEVICES

In this section, the robustness of the QCA devices and circuits has been pursued in detail. As mentioned before, the basic functionality of a QCA device is based on the Coulombic interaction among neighboring QCA cells (depending on the accuracy and geometry of its implementation). Various configurations of QCA devices have been studied using QCADesigner [3].

Recent developments in cell manufacturing (involving the deposition of molecules on a substrate surface) [4] [5] have substantially changed the nature of the QCA process fabrication. Nanometer-sized QCA cells are fabricated through a molecular implementation by a self-assembly process [6]. This QCA fabrication process has received considerable attention, resulting in very promising molecular-based devices [6]. It is anticipated that in these implementations, QCA cells (each made of two dipoles or dots) will be deposited on parallel V-shaped tracks [7]. At this level however, new types of defects (besides displacement and misalignment defects in metal QCA) are likely to occur. Missing or additional cells are inevitable for molecular implementation, because the process of cell deposition is very sensitive [5]; a small variation in process parameters may result in a defect [4]. Moreover, it will be shown that these defects[1] pronounce functional effects when they occur either within, or very near to the layout of the target device due to strong cell interactions (refer to Section 5.2.6). So, testing is required for detecting these types of defects in basic QCA devices and circuits. For molecular QCA implementations, multiple defects can be expected; however it is almost impossible to misdetect multiple defects in QCA (i.e., single fault detection is both effective and realistic).

To perform a defect characterization of QCA devices and circuits and study their effects at logic-level, appropriate defect mechanisms and models must be considered that (1) can be simulated using available simulation methods and (2) are realistic to model manufacturing and fabrication defects.

1 Commonly referred to as the deposition defects (cell displacement/misalignment, presence/absence of a cell)

Definition 5.2.1 A *cell displacement* is a defect in which the defective cell is misplaced within its original direction. Several cell displacement defects are shown in Figure 5.6.

Definition 5.2.2 In a *cell misalignment* defect, the direction of the defective cell is misplaced. Some examples of cell misalignments are shown in Figure 5.7.

Definition 5.2.3 An *extra or additional cell* (D_A) is a defect in which an additional cell is deposited at a certain location of the substrate; this extra cell is erroneously deposited along the device perimeter (adjacency boundary) of the original (defect-free) configuration.

Definition 5.2.4 In a *missing cell* deposition defect (D_M), a particular cell is missing in the original (defect-free) configuration of the device or circuit.

The defect characterization of different QCA devices in the presence of a single cell deposition defect and its effects at both device-level and circuit-level are studied in great detail. The approach proposed in this work is based on simulating deposition defects in the layout and investigating their effects at device-level to establish the functional behavior in the presence of such defects. The following defects are simulated for QCA devices: all possible combinations of cell displacement with respect to the central cell under different distances, cell misalignment in different directions and missing and extra cell defects. For QCA MV rotation is also simulated.

For D_A and D_M, injection of cell deposition defects on a Cartesian layout is performed to establish the behavior of QCA-based circuits and to generate appropriate test sets for detection. For D_A, the adjacency boundary of a cell deposition defect is considered in this chapter: the adjacency boundary consists of the area around the cell perimeter of a device or circuit in which the presence of a defect due to an additional cell deposition may occur. As interactions between QCA cells decrease with distance (at a distance x between two cells, the strength of Coulomb interactions decreases by x^{-5}), then a simple yet realistic assumption can be made in the fault model and evaluation: for deposition defects, the additional cells that have the strongest interactions are those that are adjacent to the cells in the QCA device. This set of cells defines the so-called *adjacency domain* of the device.

According to [8], in the present stage of QCA manufacturing, defects are possible in both the synthesis phase, in which the individual cells (molecules) are manufactured, and the deposition phase in which the cells are placed in a specific location on the surface. Manufacturing defects may cause a cell to have missing or extra dots and/or electrons. These defects are fatal to the correct operation of

a QCA cell and easy to detect. However, defects are much more likely to occur in the deposition process than in the synthesis process. These defects are usually categorized as cell *misplacement*. A missing dot (or additional dot) is very unlikely due to the ease of purification of small inorganic molecules [8]. For example, Nuclear Magnetic Resonance (NMR) has an estimated minimum purity of 99% for model compounds such as the Creutz-Taube (CT) Ion (a 2-dot model or dipole for half of a cell). Moreover, electrochemical measurements for the CT Ion have shown that fewer than one molecule in 10^5 are in the incorrect charge state [4]. Yet placing the individual cells during deposition is difficult and various types of cell misplacement may occur.

In this work, the behavior of a QCA device in the presence of cell deposition defects is functionally modeled into erroneous logic behavior. It will be shown that defects result in unique functional behavior. In the following sections, it will be shown that this set is given by stuck-at faults (such as S-a-A, S-a-$\bar{A}$, S-a-$\bar{B}$), different output functions (such as Maj(A', B, C')) and $undet$, where $undet$ refers to the state of undetermined QCA polarization (either extremely low polarization, or presence of glitches in a signal) and denoted by "-".

5.2.1 Simulation Engines

The bistable simulation engine of QCADesigner v.1.2.0 (Unix version) [3] is used for simulating the displacement and misalignment defects. All simulation parameters are set to default value in this engine for simulating the displacement and misalignment defects. Cell size for displacement and misalignment defect simulations is $20 \times 20nm^2$; the cell-to-cell distance and dot size are $5nm$. These parameters are chosen to be consistent with metal QCA implementation.

The coherence vector engine of QCADesigner v.1.4.0 (Unix version) is used for simulating the extra and missing cell defects; differently from metal-based QCA, in this type of implementation a defect may occur due to the erroneous deposition of cells on a substrate (i.e., missing, or an additional cell is placed either near or within the layout configuration of a QCA device). In all simulation cases of missing and extra cell defect, the radius of effect for each cell is set to $40nm$, temperature $T = 300K$, relative permittivity $\epsilon_r = 1$, clock high $ck_h = 9.8 \cdot 10^{-20}\ J$, clock low $ck_l = 3.8 \cdot 10^{-23}\ J$, and all other simulation parameters are set to the default value. The cell dimensions have been chosen according to a molecular scale as detailed in [9]: cell lateral size $d = 2.6nm$, spacing between cells $s = 0.2nm$, and dot size diameter $dot_{\oslash} = 0.6nm$. Note stable polarization results have been obtained by

simulation at room temperature. Variations in these parameters will be investigated in Section 5.2.6.

Details of the two engine were presented previously in Section 3.5.

5.2.2 MV Defect Analysis

There has been a study of the fault tolerant properties of the MV under some manufacturing misalignments [10] [11]. In this chapter, different defects in the MV (cell displacement, misalignment, extra and missing cell defects) are considered and simulated.

5.2.2.1 Cell Displacement and Misalignment Defect

The faulty results for cell displacement and misalignment are shown in Tables 5.2 and 5.3, respectively. Only faulty entries are shown in the tables.

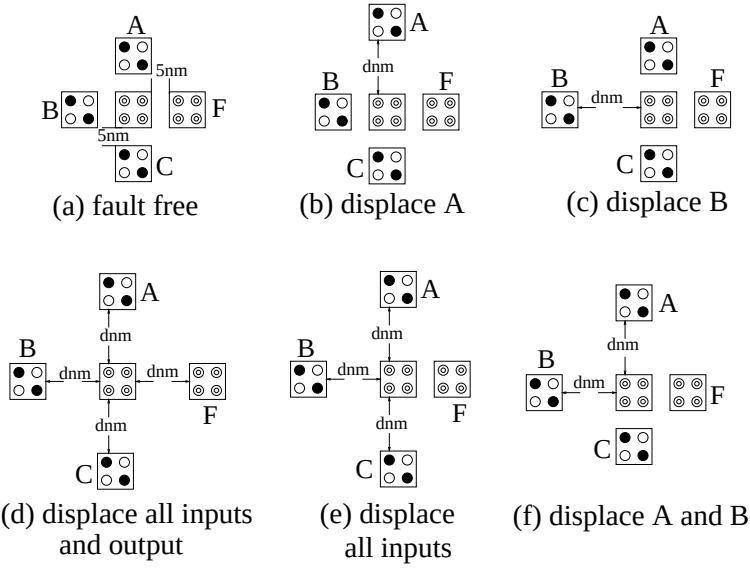


Figure 5.6 Displacement Defect in MV

The data shows that in most cases the horizontal input cell (i.e., cell B) is the dominant cell. For misalignment, any single cell misalignment greater than or

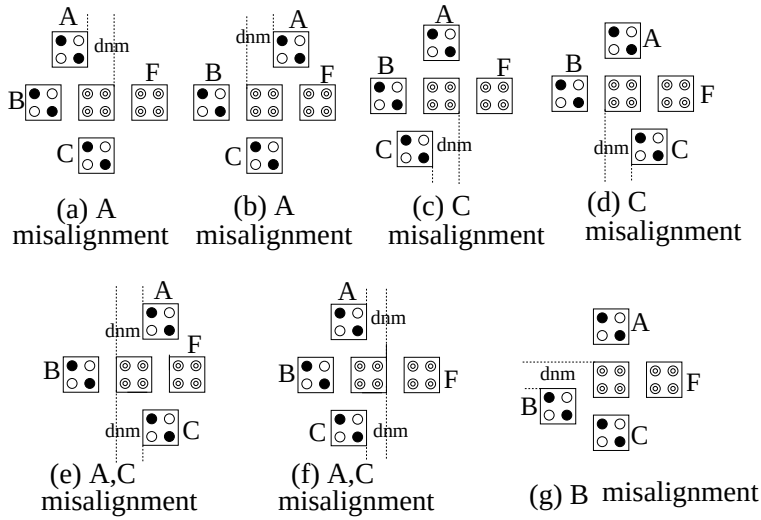


Figure 5.7 Misalignment Defect in MV

equal to half a cell causes malfunction (fault at logic-level). In some cases the error margin is even smaller.

5.2.2.2 Extra and Missing Cell Defect

Figure 5.8 shows the cell layout of the MV and the locations of the possible cell deposition defects. The x and y coordinates are used to identify the cells in the Cartesian layout. Simulation results are reported in Table 5.4 (D_M shows the coordinates of the missing cell deposition defect; D_A shows the coordinates of the extra cell deposition defect; remarkably, an extra cell deposition never affects the output as a majority function. This is applicable also to the other QCA devices as considered in next sections. The robustness of the MV in the presence of an additional cell is caused by the positive feedback to stabilize the correct polarization (as shown in [12]). As for the missing cell deposition defect, the following considerations are valid: (1) The absence of cell (2,1) or (2,3) leads in both cases to $F = B$ which confirms previous results found in misplacement simulations. (2) The absence of the middle cell (2,2) due to a missing deposition defect results in a majority function in which the input signals A and C are inverted,

that is, Maj(A', B, C'). These results have been confirmed using bistable engine and displacement/misalignment setup values.

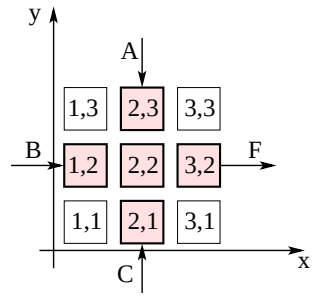


Figure 5.8 Extra and Missing Cell Defect in MV

5.2.2.3 Defect Analysis of Rotated MV

The simulation results show that MV is robust with respect to rotation of all input and output cells around the center cell, i.e., the logic-level behavior of the rotated MV is the same as the original device. Based on this observation, some simulations are performed to investigate the robustness of the Rotated MV (RMV). The basic functionality of an MV is based on the Coulombic interaction among its four neighboring input and output QCA cells, which strongly depends on the precision and geometry of its implementation. The focus is on validating different configurations of MV in the 45^o rotation, as shown in Figure 5.9.

The simulation results show that the RMV functions normally, except when moving:

- A input north, with $dA \geq 10nm$ for $ABC = 001, 110$ (the output follows the C input). A similar output appears when moving A to northeast with $dB \geq 10\sqrt{2}nm$.
- B input north, with $dB \geq 40nm$. The output is unknown (unpolarized) for $ABC = 001, 011, 100, 110$. A similar output appears when moving B to the northwest with $dB \geq 30\sqrt{2}nm$.
- C input south, with $dC \geq 15nm$ for $ABC = 011, 100$ (the output follows the A input). A similar output appears when moving C to the southwest with $dC \geq 10\sqrt{2}nm$.

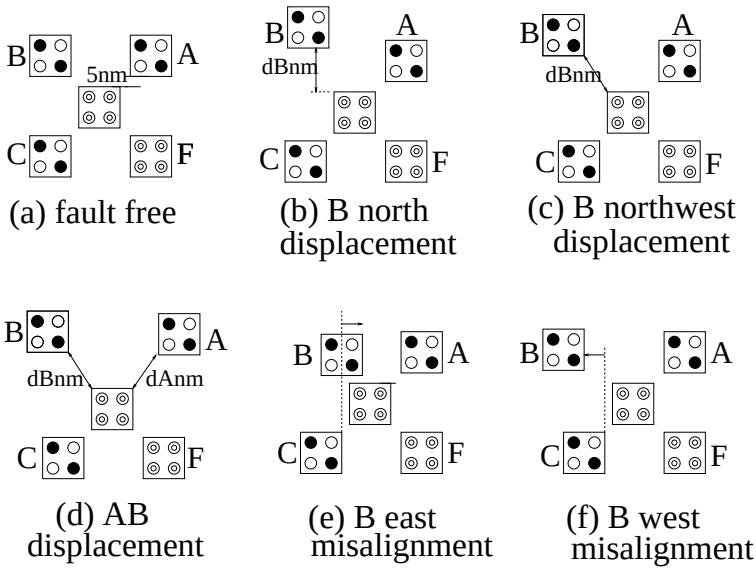


Figure 5.9 Rotated MV (Fault-Free, with Displacement or Misalignment)

- A, B, C or A, B, C, F away for $d \geq 30\sqrt{2}nm$. The output is undefined for all input combinations.
- A and B inputs away with $d \geq 10\sqrt{2}nm$ for $ABC = 001, 110$ (the output follows the C input).
- A and C inputs away with $d \geq 10\sqrt{2}nm$ for $ABC = 010, 101$ (the output follows the B input).
- B and C inputs away with $d \geq 10\sqrt{2}nm$ for $ABC = 011, 100$ (the output follows the A input).

Cell misalignment defects for RMV are also considered [e.g., Figure 5.9(e,f)]. The following shows the results for these misalignments:

- Shifting the input A west (half/full cell size), leads the output F to follow input A, while shifting A east effects the output such that it follows input C.
- RMV functions normally when input B is shifted west for a half or full cell size. However, the output is undefined for inputs $ABC = 001, 011, 100, 110$ when $dB \geq 40nm$.
- The output follows the input B, when B is shifted east for a half or full cell size.
- Similar trend is seen when input C is shifted to west or east: The output follows the input A when C is shifted west, and follows C when C is shifted east.

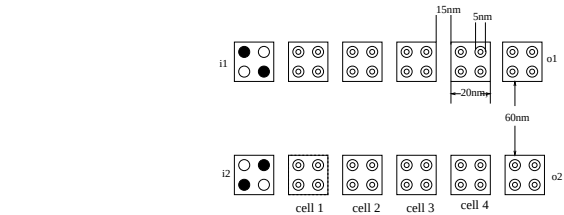
The results for different configurations of the Original MV (OMV) and the Rotated MV (RMV) are illustrated in Table 5.5. MV is completely robust with respect to rotation of all input and output cells around the central cell. This gives a significant degree of freedom for synthesizing designs based on QCA, as RMV can be used as the Original MV block. However, the original block is more dependent on the middle input (B) than the other inputs (A and C), in terms of displacement and misalignment. In the rotated version, this dependency can be completely changed based on the degree of rotation. An overall comparison in the table confirms that RMV is more fault-tolerant than the OMV. Note that only half and full misalignments are considered.

5.2.3 Interconnect Defect Analysis

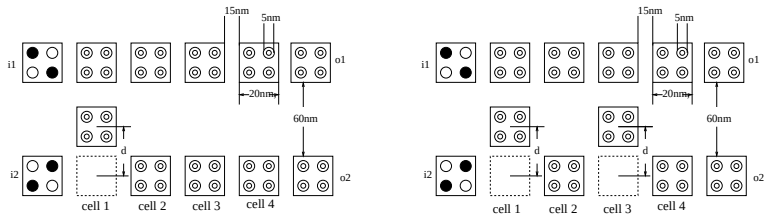
The effect of cell displacement defects on two parallel binary wires as well as two parallel inverter chains are investigated in Section 5.2.3.1. Extra and missing cell defect in Straight and L-shaped binary wires has also been investigated in Section 5.2.3.2.

5.2.3.1 Displacement Defect

Two defect-free binary wires are shown in Figure 5.10(a); the wires are denoted as the upper wire ($i1$ to $o1$) and the lower wire ($i2$ to $o2$). The cells have a size of $20 \times 20nm^2$, and the dot diameter is $5nm$. In the defect-free case, the cells in the same wire are separated by $15nm$ and the wire distance is $60nm$.



(1) Fault Free Double Wire



(2) Defects in Double Wire

Figure 5.10 Displacement in Binary Double Wires

The displacement defects are simulated by moving one or two cells in the lower wire toward the upper wire (by displacement d) as shown in Figure 5.10(b).

The simulation results are shown in Table 5.6. The results show that the upper wire is dominant in most cases: $o1$ and $o2$ are either equal to $i1$ or $i1'$, depending on which cell(s) are displaced and the value of the displacement, d. In most cases,

the upper wire functions normally (i.e., $i1 = o1$). However, in some cases the upper wire behaves as an inverter. Clearly, unlike CMOS designs, the coupling defects at QCA device-level do not behave as the *wired bridging fault* model. However, these defects manifest themselves as the dominant model (at logic level) in which the output of a wire is determined by the value of the coupled wire.

The double inverter chain is shown in Figure 5.11(a). The simulation results for moving one cell in the bottom wire toward the upper wire, with displacement d, Figure 5.11(b) are presented in Table 5.7. The displacement defects behave as the *dominating bridging fault* model at a logic level. Moreover, a comparison with the binary wires shows that binary wires are more defect tolerant than inverter chains for the case of displacement coupling defects.

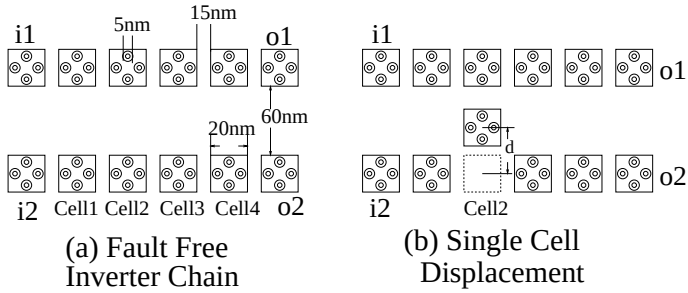


Figure 5.11 Displacement in Double Inverter Chains

5.2.3.2 Extra and Missing Cell Defect

In this subsection, simulation results for extra and missing cell defect in the wire configurations and related arrangements (straight, L-shaped, fanout and coplanar crossing) are presented.

A **straight wire** of five cell length is shown in Figure 5.12 together with the possible defect locations in the adjacency boundary. The simulation results are reported in Table 5.8. These results show that the straight wire is not sensitive to an additional cell defect as in all cases $F = A$. Moreover, also for a single missing cell deposition defect, $F = A$.

Removing a single cell from a binary wire does not affect its functionality at logic-level although it may result in some delay faults. In some cases if the cell distance is far (e.g., $15nm$ in a binary wire with $20 \times 20nm$ cell size), cell omission results in the non-conductivity of the wire.

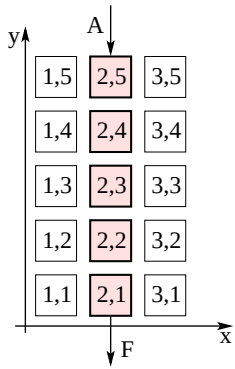


Figure 5.12 Extra and Missing Cell Defect in Straight Wire

The **L-shaped wire** is considered next; this type of wire is shown in Figure 5.13. The simulation results are reported in Table 5.9. The additional cell deposition defect does not affect the output value $F = A$, while a missing deposition defect of a cell due to an erroneous deposition has an effect only if it is the corner cell (2,2), that is, in this last case, the wire behaves as an inverter ($F = A'$).

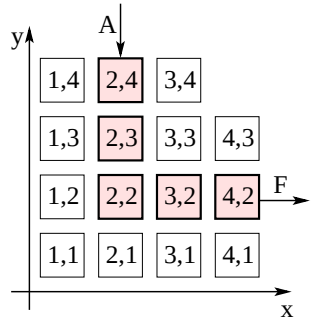


Figure 5.13 Extra and Missing Cell Defect in L-Shaped Wire

A **fanout wire** allows to duplicate a signal, so it is part of the set of basic routing devices that must be characterized. The considered layout is shown in Figure 5.14; the locations of defects are also shown. Due to symmetry, the results of Table 5.10 are valid for any cell rotation of the reported layout.

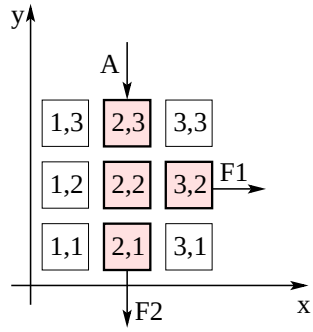


Figure 5.14 Extra and Missing Cell Defect in Fanout Wire

The results show that an extra cell deposition defect causes no functional fault in any of the output branches; instead the missing cell defect causes the output to take an undetermined value. This occurs when the cell affected by the defect is at the closest distance, i.e., cells (2,1) and (3,2); if the affected cell is the middle cell (2,2), then an inverter is formed on the path to $F1$ and therefore, an erroneous output is generated.

The last interconnect device that is considered in this section, is the so-called **coplanar crossing** of two QCA wires (as shown in Figure 5.15). The simulation results for the wire crossing device are given in Table 5.11. The extra cell deposition has been considered in both the rotated and non rotated cell arrangements. It can also be observed that a single cell omission in a wire implemented as an inverter chain results in an unwanted complementation at the output of the chain.

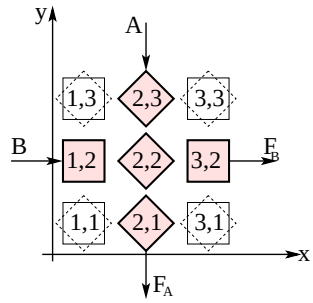


Figure 5.15 Extra and Missing Cell Defect in Coplanar Wire Crossing

From the results reported in the tables, the following considerations can be drawn: (1) Extra non rotated cells do not affect the correct outputs, while extra rotated cells affect one output only. (2) A missing cell always causes a faulty output. Moreover, two types of functional fault may occur: inversion of a signal and interference (an erroneous routing takes place in the coplanar crossing). Note that the simultaneous occurrence of these two types of fault is also possible.

In this section the robustness of QCA **inverter** device in the presence of extra and missing cell defects is investigated. Figure 5.16 shows the layout of the INV as well as the locations of possible defects; the simulation results are reported in Table 5.12. The extra cells (5,2) and (5,4) and the middle cell (4,3) change the actual layout of this device to a fanout/fanin structure, thus the output is not inverted i.e., $F = A$. For a D_M the following considerations apply: (1) A missing deposition defect at (2,3) or (2,4) or (2,2) results into $F = A$. For cell (2,3) this is a rather obvious condition, because it generates a concatenation of two inverters; for the other cells it appears that the double inverted path has a stronger effect on the output than the correct path. (2) When cell (5,3) is not deposited, the output is isolated taking an undetermined value.

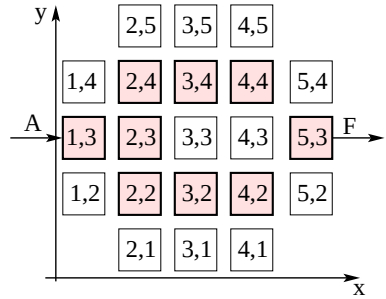


Figure 5.16 Extra and Missing Cell Defect in Inverter

5.2.4 Probabilistic Analysis and Testing

The functional behavior of the deposition defects can be used to define a probability to have a faulty output in a QCA device. For example, the results reported in the previous sections have shown that an extra cell deposition has no effect on the probability of a faulty output (except for the coplanar wire crossing). This section focuses on the molecular deposition defects (missing and additional defects).

A formal notation is introduced for defining the probability that a defective layout is generated in the manufacturing process for a non deterministic placement of QCA cells. Let $P(x, y)$ be a function that, for a given device, maps the (x, y) Cartesian coordinates of a grid layout to the probability that a cell is present in that location. Let (X, Y) denote the set of coordinate pairs of the cells that must be deposited for a desired QCA circuit layout and let P_e (P_m) denote the probability of a correctly deposited (undeposited) cell to be present (missing) at a certain location of the grid layout. Based on (x, y), $P(x, y)$ can have two different values: P_e if the cell is correctly deposited in the layout or, $(1 - P_m)$ if the cell is missing. Therefore,

$$P(x,y) = \begin{cases} P_e & \text{if } (x,y) \in (X,Y) \\ (1-P_m) & \text{if } (x,y) \notin (X,Y) \end{cases} \tag{5.1}$$

For N possible cell locations, an exponential number of layouts exists under the assumed defect model. The probability of each of these layouts is defined by the product of the probabilities (assumed to be independent) that the cells are in the locations of the layout. Therefore, for a layout of N locations, the $P(x, y)$ function defines the desired layout. If for a specific device layout, a cell is correctly deposited in the (x, y) location, then the corresponding $P(x, y)$ is P_e, else $P(x, y)$ is $1 - P_m$. As an example, consider a two-cell layout with one cell deposition shown in Fig 5.17. In this case, $N = 2$ and the values of $P(x, y)$ are given by $P(1, 1) = P_e$ for the first cell and $P(2, 1) = 1 - P_m$ for the second cell.

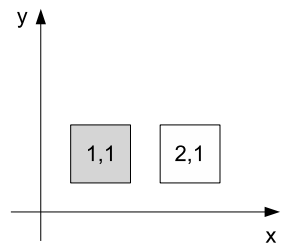


Figure 5.17 Two-cell Example

There are four possible layouts, each having the following probabilities:

- Correct layout, $P = P_e \cdot P_m$
- Defective layout with cell 1 missing, $P = (1 - P_e) \cdot P_m$

- Defective layout with additional cell 2, $P = P_e \cdot (1 - P_m)$
- Defective layout with cell 1 missing and additional cell 2, $P = (1 - P_e) \cdot (1 - P_m)$

In general, let L denote the total number of the cells in the layout, where $L = L_1 + L_2$ (L_1 is the number of cells in the device and L_2 is the number of cells in the adjacency boundary). Then, on the assumption of a single cell defect, the total probability P_T of the possible layouts (fault-free and faulty due to a single missing/extra cell defect) is:

$$P_T = P_{FF} + P_F = P_e^{L_1} \cdot P_m^{L_2} + L_1 \cdot P^{L_1 - 1} \cdot (1 - P_e) P_m^{L_2} + L_2 \cdot P_e^{L_1} \cdot P_m^{L_2 - 1} \cdot (1 - P_m)$$

Moreover, on the assumption of a single type (either rotated, or non rotated) of cell and provided $L_1 = L_{1_{FF}} + L_{1_F}$ and $L_2 = L_{2_{FF}} + L_{2_F}$ (where FF (F) denotes the fault free (faulty) layout scenario) then

$$P_{FF} = P_e^{L_1} \cdot P_m^{L_2} + L_{1_{FF}} \cdot P^{L_1 - 1} \cdot (1 - P_e) \cdot P_m^{L_2} + L_{2_{FF}} \cdot P_e^{L_1} \cdot P_m^{L_2 - 1} \cdot (1 - P_m)$$

The previously presented simulation-based characterization of defective layouts (summarized in Table 5.13) can be hereafter used to evaluate the functional faults for each device with respect to the probability of its occurrence. As shown previously, each device has a number of equivalent layouts for each functional output. Hereafter, the probabilities of the equivalent layouts are summarized to provide a grading process of the most likely functional faults for each device.

Note that it is assumed that for a given device layout, $P_e = P_m$. Moreover, as discussed previously, in a defect-free layout $P_e = P_m = 1$. P_e (and P_m) of a cell depends on the inaccuracy of the deposition process. According to the proposed approach, the probability of having a faulty output is given by the sum of the probabilities of those layouts that generate that output. Hereafter, the calculation of these probabilities is performed for each of the QCA devices and corresponding fault sets as shown in Table 5.13.

5.2.4.1 Majority Voter

The fault set of the Majority Voter is composed of three elements: $S - a - B$, $Maj(A', B, C')$, $undet$. The probability of the layouts that produce each of the faults is calculated as follows:

- $S-a-B$: this fault is generated by two faulty layouts, $D_M = (2,1),(2,3)$. Therefore, its probability is: $P_{SaB}(MV) = 2P_e^4(1-P_e)P_m^4$.
- $Maj(A',B,C')$: this fault is generated by one faulty layout $D_M = (2,2)$. Therefore, its probability is: $P_{M(A'BC')}(MV) = P_e^4(1-P_e)P_m^4$.
- $undet$: this fault is generated by one faulty layout $D_M = (3,2)$. Therefore, its probability is: $P_{undet}(MV) = P_e^4(1-P_e)P_m^4$.

5.2.4.2 Inverter

The fault set of the Inverter is composed of two elements: S-a-A and $undet$. The probability of the layouts that produce each of these faults is calculated as follows:

- $S-a-A$: this fault is generated by four faulty layouts, $D_M = (2,2),(2,3)$ and $D_A = (5,2),(5,4)$. Therefore, its probability is: $P_{SaA}(INV) = 2P_e^8(1-P_e)P_m^{12} + 2P_e^9P_m^{11}(1-P_m)$.
- $undet$: this fault is generated by one faulty layout $D_M = (5,3)$. Therefore, its probability is: $P_{undet}(INV) = 2P_e^8(1-P_e)P_m^{12}$.

5.2.4.3 Straight Wire

As shown in Table 5.13, the fault set of the Straight Wire has no elements.

5.2.4.4 L-shaped Wire

As shown in Table 5.13, the fault set of the L-shaped Wire is composed of one element: $S-a-A'$. Only one layout generates this faulty output ($D_M = (2,2)$). Therefore, the probability is: $P_{SaA}(LWire) = P_e^4(1-P_e)P_m^{10}$.

5.2.4.5 Fanout

The fault set of the Fanout is composed of two elements on F1 ($S-a-A'$ and $undet$) and of one element on F2 ($undet$). The probability of the layouts that produce the faults on F1 is calculated as follows:

- $S-a-A'$: this fault is generated by one faulty layout, $D_M = (2,2)$. Therefore, its probability is: $P_{SaA}(Fanout1) = P_e^3(1-P_e)P_m^5$.
- $undet$: this fault is generated by one faulty layout $D_M = (3,2)$. Therefore, its probability is: $P_{undet}(Fanout1) = P_e^3(1-P_e)P_m^5$.

The probability of the layouts that produce the fault on F2, is calculated as follows:

- $undet$: this fault is generated by one faulty layout $D_M = (2, 1)$. Therefore, its probability is: $P_{undet}(Fanout2) = P_e^3(1 - P_e)P_m^5$.

5.2.4.6 Coplanar Wire Crossing

The fault set of the Coplanar Wire Crossing is composed of one element on F_A ($S - a - A'$) and two elements on F_B ($S - a - A'$ $undet$). The probability of the layouts that produce the fault on F_A is calculated as follows:

- $S - a - A'$: this fault is generated by three faulty layouts, $D_M = (2, 1)(2, 2)(2, 3)$. Therefore, its probability is: $P_{SaA'}(crossF_A) = 3P_e^4(1 - P_e)P_m^4$.

The probability of the layouts which produce the faults on F_B, is calculated as follows:

- $S - a - A'$: this fault is generated by two faulty layouts, $D_M = (1, 2)$ and $D_A = (3, 3)$ (with a non rotated cell). Therefore, its probability is: $P_{SaA'}(crossF_B) = P_e^4(1 - P_e)P_m^4 + P_e^5P_m^3(1 - P_m)$.
- $undet$: this fault is generated by one faulty layout $D_M = (2, 1)$. Therefore, its probability is: $P_{undet}(crossF_B) = P_e^4(1 - P_e)P_m^4$.

Test selection by grading can be established based on this analysis. By considering that all possible faults are functions of P_e and P_m and assuming that $P_e = P_m = P$, all above reported probabilities can be expressed as $P_i(j) = f(P)$ where $i \in \{undet, S - a - A', S - a - A, M(A', B, C'), S - a - B\}, j \in \{MV, INV, LWire, Fanout1, Fanout2, crossF_A, crossF_B\}$. For a given value of P, grading of the most likely faults can be obtained. So, a weight (which takes a value in the range of 0 to 1 where 1 is associated with the fault of the highest probability) can be introduced for each possible fault; such weight is therefore defined as

$$W_{ij}(P) = \frac{P_i(j)(P)}{P_{MAX}(P)} \tag{5.2}$$

where $P_{MAX}(P)$ is the highest probability for the faults in each pair (i, j) at a given probability P.

The above fault analysis can be utilized for generating test vectors through a weighted (grading) approach. For CMOS circuits, test vector generation has been extensively analyzed; heuristic criteria and related techniques (such as through the

use of a fault dictionary and grading) have been introduced to reduce the time complexity involved in Automatic Test Pattern Generation (ATPG). For QCA, no ATPG is currently available and test generation poses different issues than CMOS; the presence of non classical faults (beyond the stuck-at, or bridge fault models of VLSI) and the geometric implications on the correct operation of devices and circuits necessitate a different metric for grading.

A greedy (heuristic) approach based on this metric is proposed in this chapter, namely to utilize the weight as criterion for selecting and prioritizing vectors when testing molecular QCA. The proposed test generation approach can be briefly described as follows: after sorting all functional fault sites (as per the molecular QCA defect model presented previously), test vectors are generated according to a descending order of weight, thus testing high likely functional faults early in the testing process. Other features (such as observability/controllability of the QCA circuit, redundant faults and collapsing) are also considered. This process is iteratively executed until either all possible faults are tested by the generated set, or the desired weighted coverage has been achieved. For a test vector t the weighted coverage is defined as

$$W_C(t) = \frac{\sum_{k=1}^{d} W_{ij}(k)}{\sum_{k=1}^{N} W_{ij}(k)} \tag{5.3}$$

where d is the number of faults detected by the considered vector, N is the total number of faults in the circuit and W_{ij} is the weight as defined above for a given P of the generic (i, j) fault in the circuit. The above reported definition of fault coverage is used throughout this chapter as opposed to the unweighted figure, commonly given by

$$F_C(t) = \frac{d}{N} \tag{5.4}$$

An unweighted coverage of less than 100% represents a condition that much likely will be encountered in practice [5] due to the expected large number of QCA cells in molecular implementations; as an extremely high complexity will be encountered in these circuits, a weighted test generation procedure will cover the most likely faults.

5.2.5 Defect Analysis and Testing of QCA Circuits

The previously described approach has been applied to generate the test vectors for four QCA circuits: a two-input XOR gate, a full adder and a 2-to-4 decoder.

Initially, the assumption of single fault occurrence per QCA circuit is upheld. For each of the considered circuits, the following sections provide the test vectors and the simulation results inclusive of the corresponding (fault-free and faulty) outputs for each defect at a specified location.

This data has been validated by a preliminary step in which injection of single missing or additional deposition defect has been performed on the QCA circuit and the value at the primary outputs (PO) has been compared to the one obtained by combining the effects of the fault set on each of the devices in the circuit. This process can be used to validate and confirm that the device-level analysis presented in previous sections can be extended to circuit-level, while obtaining consistent results for the functional fault set. In this respect, full validation has been accomplished; an interesting effect that has been captured for molecular QCA, is the undetermined fault. These faults can be propagated and detected only when the output of the affected device is a primary output: when a device that is affected by this type of fault is located internally to the QCA circuit, then the undetermined value never propagates and the correct output is always observed at the primary output. This is caused by the regenerative effect of the non-linear nature of the cell-to-cell QCA response, as a weak polarization appears only at an isolated output; however, when the output is not isolated, cell-to-cell interaction causes the regeneration of the weak polarization and therefore, propagation of the functional fault does not occur. An undermined fault that is not propagated is denoted as NP_{undet}. Note that, for testing purposes, the undetermined faults have been considered always undetectable if internal to the circuit and always detectable if at a primary output.

5.2.5.1 EXOR Gate

The QCA schematic diagram of the considered EXOR gate circuit is shown in Figure 5.18.

Table 5.14 shows all functional faults affecting the QCA devices in the EXOR (given in schematic form in Figure 5.19). Test vectors are reported; specifically, the undetermined fault at MV3 is labelled as non propagating undetermined fault (denoted as NPC_{undet}). This refers to the condition whether MV3 provides the PO of the QCA circuit. If so, then the undetermined fault is detected; otherwise, this fault is NP_{undet} (for example a QCA wire is placed after the output of MV3).

For the EXOR gate circuit, only two test vectors are required for 100% single fault coverage (i.e., 00 and 11). Table 5.15 shows the coverage and the normalized weight of each test vector. Using the analysis previously presented, a traditional

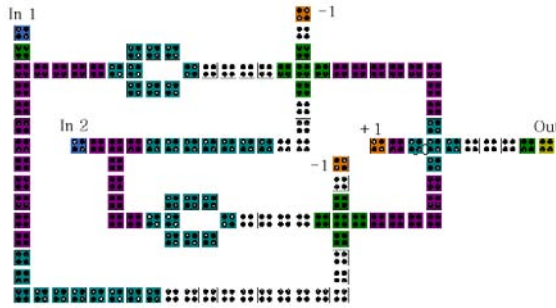


Figure 5.18 EXOR Gate Circuit, QCA Layout

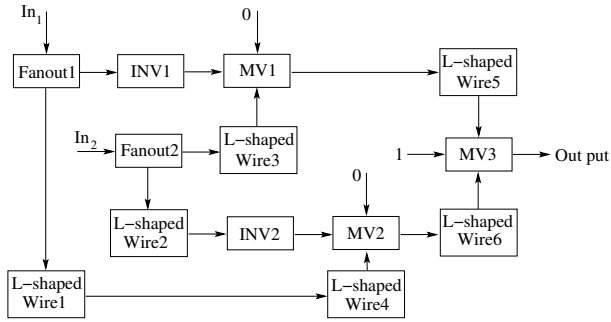


Figure 5.19 EXOR Gate Circuit, Device Level Schematic Diagram

analysis of coverage (the percentage of possible faults detected by the vectors over the total number of faults possible in the circuit) yields 48% and 40% for vectors 00 and 11, respectively. By considering the proposed weight, vectors 00 and 11 account for 51.5% and 40.09% (i.e., a lower weighted coverage). This discrepancy is attributed to the unique nature of faults and defects in QCA and the inability of a traditional coverage calculation to precisely establish such figures of merit under a more complex defect model (as applicable to deposition defects in molecular QCA). Such difference is also encountered in the test set: two vectors are needed for detecting a single QCA deposition defect and induced faults for a molecular implementation. An EXOR gate implemented in VLSI requires three test vectors under a single stuck-at/bridge fault model at the primary input/output lines (i.e., 01, 11 and 00) and four test vectors (01, 10, 11, 00) under a single unrestricted combinational fault model.

5.2.5.2 Full Adder

The test generation approach has been applied to a full adder circuit, whose QCA schematic diagram is shown in Figure 5.20.

Table 5.16 shows all functional faults affecting the QCA devices in the full adder (given in schematic form in Figure 5.21). A missing cell defect was injected in all devices, except straight wires. For additional cell defects, only INVs and MVs were considered because as shown previously, these defects have limited effects on a QCA interconnect (as wires). Also, missing defects on the cells at the corner and next to the corner of the straight wires were tested. For testing the full adder circuit under this model at 100% coverage, only two test vectors are required, that is, any set in $\{abc\} = \{010, 101\} \times \{100, 001\}$, where the operator $\times$ denotes the Cartesian product. For example, the test set $\{010, 100\}$ is one of the minimal test sets. Table 5.17 shows the coverage and the normalized weight of each test vector.

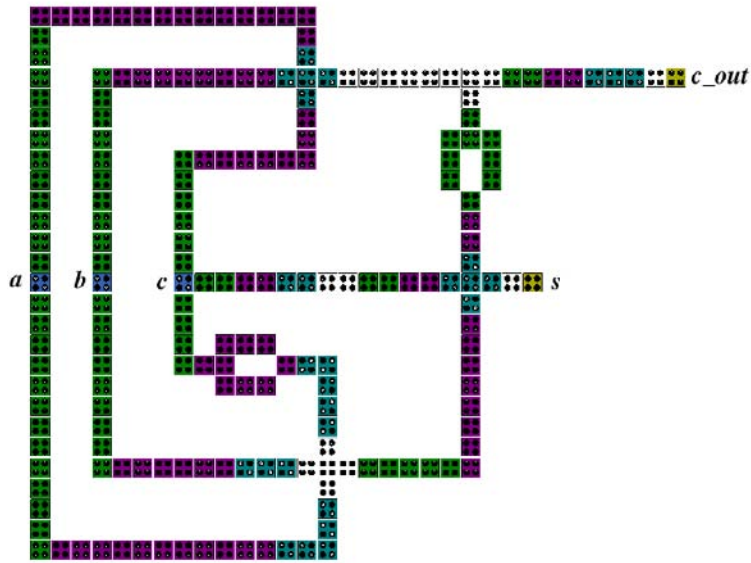


Figure 5.20 Full Adder Circuit, QCA Layout

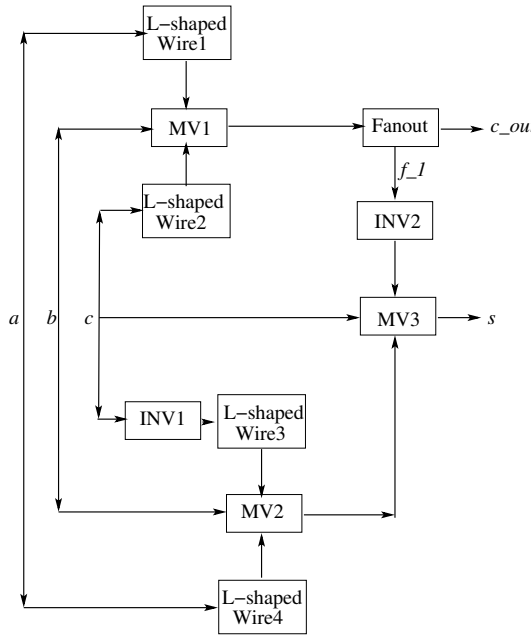


Figure 5.21 Full Adder, Device Level Schematic Diagram

Table 5.2
Results for displacement in MV

<table>
<tr><td colspan="3">displace cell A: Figure 5.6(b)</td></tr>
<tr><td colspan="2">$d \leq 15nm$ Normal Operation</td><td>$d \geq 20nm$, $F = B$</td></tr>
<tr><td colspan="3">displace cell B: Figure 5.6(c)</td></tr>
<tr><td colspan="2" rowspan="6">$d \leq 40nm$
Normal Operation</td><td colspan="2">$d \geq 45nm$</td></tr>
<tr><td>ABC</td><td>F</td></tr>
<tr><td>001</td><td>Z (no polarization)</td></tr>
<tr><td>011</td><td>Z(no polarization)</td></tr>
<tr><td>100</td><td>Z (no polarization)</td></tr>
<tr><td>110</td><td>Z (no polarization)</td></tr>
<tr><td colspan="3">displace all input/output cells: Figure 5.6(d)</td></tr>
<tr><td colspan="2" rowspan="3">$d \leq 10$ or $30 \leq d \leq 40nm$
Normal Operation
$d \geq 45nm$
$F = Z$ (no polarization)</td><td colspan="2">$15 \leq d \leq 25nm$</td></tr>
<tr><td>ABC</td><td>F</td></tr>
<tr><td>010
101</td><td>0/1
1/0</td></tr>
<tr><td colspan="3">displace all input cells: Figure 5.6(e)</td></tr>
<tr><td colspan="2">$d \leq 15$ or $d = 40nm$
Normal Operation</td><td colspan="2">$d \geq 45nm$
$F = Z$ (no polarization)</td></tr>
<tr><td colspan="2">$20 \leq d \leq 25$ or $d = 35nm$</td><td colspan="2">$d = 30nm$</td></tr>
<tr><td>ABC</td><td>F</td><td>ABC</td><td>F</td></tr>
<tr><td>010</td><td>0/1</td><td>000</td><td>0/1</td></tr>
<tr><td>101</td><td>1/0</td><td>010</td><td>0/1</td></tr>
<tr><td></td><td></td><td>101</td><td>1/0</td></tr>
<tr><td></td><td></td><td>111</td><td>1/0</td></tr>
<tr><td colspan="3">displace cells A and B: Figure 5.6(f)</td></tr>
<tr><td colspan="2">$d \leq 5nm$ Normal Operation</td><td>$d \geq 10nm$, $F = C$</td></tr>
</table>

Table 5.3
Results for Misalignment in MV

<table>
<tr><td colspan="3">move A toward west: Figure 5.7(a)</td></tr>
<tr><td colspan="2">$d \leq 5nm$ Normal Operation</td><td>$d \geq 10nm$, $F = B$</td></tr>
<tr><td colspan="3">move A toward east: Figure 5.7(b)</td></tr>
<tr><td colspan="2">$5 \leq d \leq 15nm$</td><td rowspan="2">$d = 20$ or $d = 30nm$
Normal Operation</td></tr>
<tr><td>ABC</td><td>F</td></tr>
<tr><td>001</td><td>0/1</td><td rowspan="4">$d = 25nm$
$F = A$</td></tr>
<tr><td>010</td><td>0/1</td></tr>
<tr><td>101</td><td>1/0</td></tr>
<tr><td>110</td><td>1/0</td></tr>
<tr><td colspan="3">move C toward west: Figure 5.7(c)</td></tr>
<tr><td colspan="2">$d \leq 5nm$
Normal Operation</td><td>$d \geq 10nm$
$F = B$</td></tr>
<tr><td colspan="3">move C toward east: Figure 5.7(d)</td></tr>
<tr><td colspan="2">$5 \leq d \leq 15nm$</td><td rowspan="2">$d = 20$ or $d = 30nm$
Normal Operation</td></tr>
<tr><td>ABC</td><td>F</td></tr>
<tr><td>010</td><td>0/1</td><td rowspan="4">$d = 25nm$
$F = C$</td></tr>
<tr><td>011</td><td>1/0</td></tr>
<tr><td>100</td><td>0/1</td></tr>
<tr><td>101</td><td>1/0</td></tr>
<tr><td colspan="3">move A, C toward west: Figure 5.7(e)</td></tr>
<tr><td colspan="2">$d \geq 5nm$, $F = B$</td><td></td></tr>
<tr><td colspan="3">move A, C toward east: Figure 5.7(f)</td></tr>
<tr><td colspan="2">$d = 5, 20$, $d \geq 30nm$
$F = B$</td><td colspan="2">$10nm \leq d \leq 15nm$</td></tr>
<tr><td colspan="2" rowspan="2">$d = 25nm$
Normal Operation</td><td>ABC</td><td>F</td></tr>
<tr><td>000</td><td>0/1</td></tr>
<tr><td colspan="2" rowspan="3"></td><td>010</td><td>0/1</td></tr>
<tr><td>101</td><td>1/0</td></tr>
<tr><td>111</td><td>1/0</td></tr>
<tr><td colspan="3">move B toward south/north: Figure 5.7(g)</td></tr>
<tr><td colspan="2" rowspan="2">$d \leq 5nm$
Normal Operation</td><td colspan="2">$d \geq 45nm$</td></tr>
<tr><td>ABC</td><td>F</td></tr>
<tr><td colspan="2" rowspan="4"></td><td>001</td><td>0/1</td></tr>
<tr><td>011</td><td>1/0</td></tr>
<tr><td>100</td><td>0/1</td></tr>
<tr><td>110</td><td>1/0</td></tr>
</table>

Table 5.4
Simulation results for MV

D_M	F	D_A	F
2,1	B	1,1	Maj(A, B, C)
1,2	Maj(A, B, C)	1,3	Maj(A, B, C)
2,3	B	3,1	Maj(A, B, C)
2,2	Maj(A', B, C')	3,1	Maj(A, B, C)
3,2	-		

Table 5.5
Original MV vs. Rotated MV

Config.	Faults	OMV	RMV
A move	distance	$d \geq 20nm$	$d \geq 10$(N) or $10\sqrt{2}nm$ (NE)
	# of faults	2	2
B move	distance	$d \geq 45nm$	$d \geq 40$(W) or $30\sqrt{2}nm$ (NW)
	# of faults	4	4
C move	distance	$d \geq 20nm$	$d \geq 10$(S) or $10\sqrt{2}nm$ (SW)
	# of faults	2	2
ABC move	distance	$20 \leq d \leq 35$ or $d \geq 45nm$	$d \geq 30\sqrt{2}nm$
	# of faults	2/4/8	8
$ABCF$ move	distance	$15 \leq d \leq 25$ or $d \geq 45nm$	$d \geq 30\sqrt{2}nm$
	# of faults	2/8	8
AB move	distance	$d \geq 7.5nm$	$d \geq 10\sqrt{2}nm$
	# of faults	2	2
AC move	distance	$d \geq 7.5nm$	$d \geq 10\sqrt{2}nm$
	# of faults	2	2
F move	distance	$d \geq 45nm$	$d \geq 30\sqrt{2}nm$
	# of faults	8	8
AC misalignment	# of faults	4	4
B misalign. West	# of faults	4	0
B misalign. East	# of faults	4	2

Table 5.6

Displacement Results for Double Binary Wires

move cell1 OR cell2		
$d \leq 40nm$	$d = 45 - 50nm$	$d \geq 55nm$
Normal	$o1 = i1, o2 = i1$	$o1 = i1, o2 = Z$

move cell3 OR cell4		
$d \leq 35nm$	$d = 40 - 50nm$	$d \geq 55nm$
Normal	$o1 = i1, o2 = i1'$	$o1 = i1, o2 = Z$

move cell1 AND cell2		
$d \leq 35nm$	$d = 40 - 50nm$	$d \geq 55nm$
Normal	$o1 = i1, o2 = i1$	$o1 = i1, o2 = Z$

move cell1 AND cell4; OR move cell 2 AND cell 3; OR move cell3 AND cell4		
$d \leq 35nm$	$d = 40 - 50nm$	$d \geq 55nm$
Normal	$o1 = i1, o2 = i1'$	$o1 = i1, o2 = Z$

move cell1 AND cell3			
$d \leq 35nm$	$d = 40 - 50nm$	$d = 45nm$	$d \geq 55nm$
Normal	$o1 = i1, o2 = i1$	$o1 = i1, o2 = i1'$	$o1 = i1, o2 = Z$

move cell2 AND cell4				
$d \leq 15nm$	d=20-25nm d=40-45nm	d=30-35nm	$d = 50nm$	$d \geq 55nm$
Normal	$o1 = i1$	$o1 = i1$ $o2 = i1$	$o1 = i1'$ $o2 = i1$	$o1 = i1$ $o2 = Z$

Table 5.7

Displacement Results for Double Inverter Chains

Fault Free: $o1 = i1'; o2 = i2'$		
move cell1 OR cell2 OR cell3		
$d \leq 35nm$	$d = 40nm - 50nm$	$d \geq 55nm$
Normal	$o1 = i1', o2 = i1'$	$o1 = i1', o2 = Z$
move cell4		
$d \leq 30nm$	$d = 35nm - 50nm$	$d \geq 55nm$
Normal	$o1 = i1', o2 = i1'$	$o1 = i1', o2 = Z$

Table 5.8

Simulation Results for Straight Wire

D_M	F	D_A	F	D_A	F
2,5	A	3,5	A	1,5	A
2,4	A	3,4	A	1,4	A
2,3	A	3,3	A	1,3	A
2,2	A	3,2	A	1,2	A
2,1	A	3,1	A	1,1	A

Table 5.9

Simulation Results for L-shaped Wire

D_M	F	D_A	F	D_A	F
2,4	A	1,4	A	3,1	A
2,3	A	1,3	A	4,1	A
2,2	A'	1,2	A	3,4	A
3,2	A	1,1	A	3,3	A
4,2	A	2,1	A	4,3	A

Table 5.10

Simulation Results for Fanout Wire

D_M	$F1$	$F2$	D_A	$F1$	$F2$
2,1	A	-	3,1	A	A
2,2	A'	A	1,2	A	A
3,2	-	A	1,3	A	A
2,3	A	A	3,3	A	A

Table 5.11

Simulation Result for Coplanar Wire Crossing

D_M	F_A	F_B	D_A, Non rotated Cell	F_A	F_B	D_A, Rotated Cell	F_A	F_B
2,1	-	B	1,1	A	B	1,1	A	B
1,2	A	A'	3,1	A	B	3,1	A	B
2,2	A'	B	1,3	A	B	1,3	A	B
3,2	A	-	3,3	A	A'	3,3	A	B
2,3	A'	B						

Table 5.12
Simulation Results for Inverter

D_M	F	D_A	F
2,2	A	2,1	A'
3,2	A'	3,1	A'
4,2	A'	4,1	A'
1,3	A'	1,2	A'
2,3	A	5,2	A
5,3	-	3,3	A'
2,4	A	4,3	A
3,4	A'	1,4	A'
4,4	A'	5,4	A
		2,5	A'
		3,5	A'
		4,5	A'

Table 5.13
Fault Set for QCA Devices with Single Cell Defect

Device	Fault Set
MV	S-a-B $Maj(A', B, C')$ undet
INV	S-a-A undet
Straight wire	none
L-shaped wire	S-a-A'
Fanout	(F1) S-a-A' (F1) undet (F2) undet
Coplanar Wire Crossing	(F_A) S-a-A' (F_B) S-a-A' (interference) (F_B) undet

Table 5.14
Test Vectors for EXOR Gate Circuit

Fault Site	**Fault**	**Test Vector** In_1In_2	**Primary Output** fault-free (faulty)
MV1&MV2	S-a-B	00	0(1)
	$Maj(A', B, C')$	00	0(1)
	undet	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
MV3	S-a-B	00,11	0(1)
	$Maj(A', B, C')$	00 ,11	0(1)
	undet	NPC_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
INV1	S-a-A	$In_1$1	$In_1'(In_1)$
	undet	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
INV2	S-a-A	1In_2	$In_2'(In_2)$
	undet	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
Fanout1	s-a-A' for f_1	$In_1$1	$In_1'(In_1)$
	undet f_1	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
	undet f_1	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
Fanout2	S-a-A' for f_1	1In_2	$In_2'(In_2)$
	undet f_1	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
	undet f_2	NP_{undet}	$In_1 \oplus In_2$ ($In_1 \oplus In_2$)
L-shaped wire1	S-a-A'	$In_1$0	$In_1(In_1')$
L-shaped wire2	S-a-A'	1In_2	$In_2'(In_2)$
L-shaped wire3	S-a-A'	0In_2	$In_2(In_2')$
L-shaped wire4	S-a-A'	$In_1$0	$In_1(In_1')$
L-shaped wire5	S-a-A'	$In_1$1, 00	$In_1'(In_1)$, 0(1)
L-shaped wire6	S-a-A'	1In_2, 00	$In_2'(In_2)$, 0(1)

Table 5.15
Coverage and Weight Comparison of the Test Vectors for the EXOR Gate Circuit

Test Vector	F_C **(%)**	W_C **(%)**
00	48	51.5
11	40	40.09

Table 5.16 Defects and Test Vectors for Full Adder Circuit

Fault Site	**Defective Cell** Missing(m), Add(a)	**Fault**	**Test Vector** $abcsel_1$	**Primary Output** fault-free (faulty)
MV1	m2,3;m2,1	S-a-B	010,101	s = 1(0) c_out = 0(1),s = 0(1) c_out = 1(0)
	m2,2	$Maj(A', B, C')$	000,010,101,111	c_out = 0(1),s = 1(0) c_out = 0(1), s = 0(1) c_out = 1(0),c_out = 1(0)
	m3,2	undet	NP_{undet}	
MV2	m2,3;m2,1	S-a-B	011,100	s = 0(1),1(0)
	m2,2	$Maj(A', B, C')$	011,100	s = 0(1),1(0)
	m3,2	undet	NP_{undet}	
MV3	m2,3;m2,1	S-a-B	010,011,100,101	s = 1(0),0(1),1(0),0(1)
	m2,2	$Maj(A', B, C')$	010,011,100,101	s = 1(0),0(1),1(0),0(1)
	m3,2	undet	NP_{undet}	
INV1	m2,2;m2,3;m2,4 a4,3;a5,4;a5,2	S-a-A	010,011,100,101	s = 1(0),0(1),1(0),0(1)
	m5,3	undet	NP_{undet}	
INV2	m2,2;m2,3;m2,4 a4,3;a5,4;a5,2	S-a-A	001,010,011,100,101,110	s = 1(0),1(0),0(1),1(0),0(1),0(1)
	m5,3	undet	NP_{undet}	
Fanout	m2,2	S-a-A' for f_1	001,010,011,100,101,110	s = 1(0),1(0),0(1),1(0),0(1),0(1)
	m3,2	undet f_1	NP_{undet}	
	m2,1	undet c_{out}	NPC_{undet}	
L-shaped wire1	m_corner cell	S-a-A'	001,010,101,110	s = 1(0) c_out = 0(1),s = 1(0) c_out = 0(1), s = 0(1) c_out = 1(0),s = 0(1) c_out = 1(0)
L-shaped wire2	m_corner cell	S-a-A'	010,011,100,101	s = 1(0) c_out = 0(1),s = 0(1) c_out = 1(0), s = 1(0) c_out = 0(1),s = 0(1) c_out = 1(0)
L-shaped wire3	m_corner cell	S-a-A'	010,011,100,101	s = 1(0),0(1),1(0),0(1)
L-shaped wire4	m_corner cell	S-a-A'	000,010,011,100,101,110	s = 1(0),1(0),0(1),1(0),0(1),0(1)

Table 5.17
Coverage and Weight Comparison of Test Vectors for Full Adder Circuit

Test Vector	F_C **(%)**	W_C **(%)**
010	12/20 (60)	60
101	12/20 (60)	60
100	11/20 (57)	55
011	11/20 (57)	55

5.2.5.3 2-to-4 Decoder

The 2-to-4 decoder circuit whose QCA schematic diagram is shown in Figure 5.22 has also been considered. All functional faults affecting the QCA devices in the decoder (given in schematic form in Figure 5.23) are given in Table 5.18. Injection of missing cell defects has been performed on all devices except straight wires. For additional cell defects, injection only on INV devices has been performed. For testing the decoder circuit, only three test vectors are required, i.e., $sel_0 sel_1 = \{01, 10, 11\}$. Table 5.19 shows the coverage and the normalized weight of each test vector.

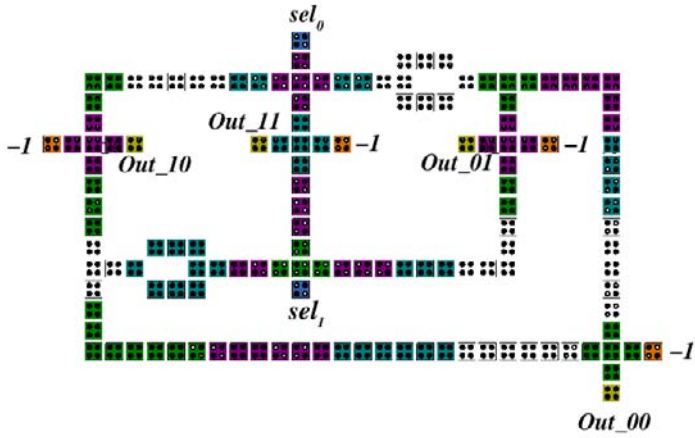


Figure 5.22 2-to-4 Decoder Circuit, QCA Layout

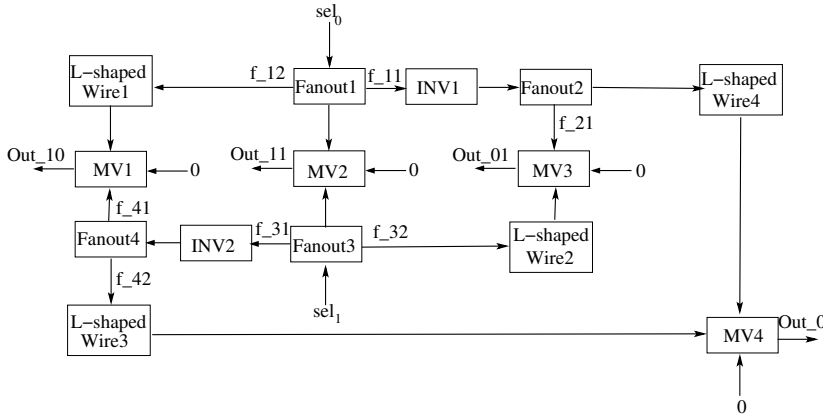


Figure 5.23 2-to-4 Decoder, Device Level Schematic Diagram

In all considered QCA circuits, 100% fault coverage has been therefore achieved under the proposed fault model (consisting of a single defect) using a minimal test set. Single defect occurrence has been assumed in each QCA circuit. Hereafter, this assumption is modified and the analysis in the presence of two defective devices (each with a single defect) is pursued for a circuit. The analysis deals with masking due to two defective devices, as tested by utilizing either the exhaustive test set, or the minimal test set for detection. Therefore, for the defects occurring in two devices of the same circuit, masking is said to occur if there is no input vector for which an output is different from the fault free one. Table 5.20 shows the defects in the two devices for which masking occurs in the considered circuits. Note that in all cases, masking does not occur when one of the two faulty devices is the Majority Voter.

An evaluation of the fault coverage has also been pursued; fault coverage is defined as the percentage of two defects (one per device) detected by a test set with respect to the total number of defect pairs. Table 5.21 shows the fault coverage of an exhaustive test set for detecting defects in two devices of each circuit (i.e., one defect per device). Fault coverage is also reported for the minimal test set. The results are summarized in Table 5.21. For example, the minimal test set for the full adder ($\{abc\} = \{010, 101\} \times \{100, 001\}$) does not detect two defects in the following devices: Fanout and INV2, INV1 and LS Wire3, INV1 and LS Wire5, LS Wire3 and LS Wire5. Hence, the fault coverage of this minimal test set under the assumed fault model is $71/75 = 94.6\%$. As mentioned previously, the simulation results show that the minimal test set for the 2-to-4 decoder is $sel_0 sel_1 = (01, 10, 11)$. The fault coverage for detection of two defects (one defect per device) remains the same as for the exhaustive test set (i.e., 99%). So while 100% coverage is possible in a device using a minimal test set, masking occurs if two devices are faulty. thus resulting in a degradation of coverage for both the fully exhaustive and minimal test sets.

5.2.6 Scaling in the Presence of Defects

In this section, we evaluate scaling of QCA devices MV and INV, in the presence of displacement and misalignment defects. The two engines of QCADesigner v.1.4.0 (Bistable and Coherence vector engine) are employed to simulate QCA devices and report the defect tolerance. In all cases, the number of simulations in the bistable engine is given by 6400; the barrier for clock low (high) is set to 3.8×10^{-23} (9.8×10^{-21}) J and $R = 4 \times l$. All other parameters are set to the default value.

Table 5.18 Defects and Test Vectors for Decoder Circuit

Fault Site	**Defective Cell** Missing(m), Add(a)	**Fault**	**Test Vector** $sel_0 sel_1$	**Primary Output** fault-free (faulty)
MV1	m2,3;m2,1	S-a-B	10, 01	$Out\_10 = 1(0)$
	m2,2	$Maj(A', B, C')$	01, 10	$Out\_10 = 0(1)$, 1(0)
	m3,2	undet	NPC_{undet}	
MV2	m2,3;m2,1	S-a-B	11	$Out\_11 = 1(0)$
	m2,2	$Maj(A', B, C')$	00, 11	$Out\_11 = 0(1)$, 1(0)
	m3,2	undet	NPC_{undet}	
MV3	m2,3;m2,1	S-a-B	01	$Out\_01 = 1(0)$
	m2,2	$Maj(A', B, C')$	10, 01	$Out\_01 = 0(1)$, 1(0)
	m3,2	undet	NPC_{undet}	
MV4	m2,3;m2,1	S-a-B	10	$Out\_00 = 0(1)$
	m2,2	$Maj(A', B, C')$	$1sel_1$	$Out\_00 = 0(1)$
	m3,2	undet	NPC_{undet}	
INV1	m2,2;m2,3;m2,4 a4,3;a5,4;a5,2	S-a-A	$sel_0 1$, $sel_0 0$	$Out\_01 = sel'_0(sel_0)$, $Out\_00 = sel'_0(sel_0)$
	m5,3	undet	NP_{undet}	
INV2	m2,2;m2,3;m2,4 a4,3;a5,4;a5,2	S-a-A	$1sel_1, 0sel_1$	$Out\_10 = sel'_1(sel_1)$, $Out\_00 = sel'_1(sel_1)$
	m5,3	undet	NP_{undet}	
Fanout1	m2,2	S-a-A' for f_11 and f_12	$sel_0 1$, $sel_0 0$	$Out\_01 = sel'_0(sel_0)$, $Out\_00 = sel'_0(sel_0)$ $Out\_10 = sel_0(sel'_0)$
	m3,2	undet f_1	NP_{undet}	
	m2,1	undet f_2	NP_{undet}	
Fanout2	m2,2	S-a-A' for f_21	$sel_0 1$	$Out\_10 = sel'_0(sel_0)$
	m3,2	undet f_1	NP_{undet}	
	m2,1	undet f_2	NP_{undet}	
Fanout3	m2,2	S-a-A' for f_31 and f_32	$1sel_1, 0sel_1$	$Out\_01 = sel'_1(sel_1)$, $Out\_00 = sel'_1(sel_1)$ $Out\_01 = sel_1(sel'_1)$
	m3,2	undet f_1	NP_{undet}	
	m2,1	undet f_2	NP_{undet}	
Fanout4	m2,2	S-a-A' for f_41 and f_42	$1sel_1, 0sel_1$	$Out\_10 = sel'_1(sel_1)$, $Out\_00 = sel'_1(sel_1)$
	m3,2	undet f_1	NP_{undet}	
	m2,1	undet f_2	NP_{undet}	
L-shaped wire1	m_corner cell	S-a-A'	$sel_0 0$	$Out\_10 = sel_0(sel'_0)$
L-shaped wire2	m_corner cell	S-a-A'	$0sel_1$	$Out\_01 = sel_1(sel'_1)$
L-shaped wire3	m_corner cell	S-a-A'	$0sel_1$	$Out\_00 = sel'_1(sel_1)$
L-shaped wire4	m_corner cell	S-a-A'	$sel_0 0$	$Out\_00 = sel'_0(sel_0)$

Table 5.19

Coverage and Weight Comparison of the Test Vectors for the Decoder Circuit

Test Vector	F_C **(%)**	W_C **(%)**
01	16/28 (51.6)	57.1
10	16/28 (51.6)	57.1
11	13/28 (42.5)	46.4

Table 5.20

Masking in the Presence of Two Defective Devices (One Defect per Device)

Circuit	**Masked Faults in Devices**
EXOR	INV1(S-a-A) & Fanout1 (S-a-A'of f1) INV2(S-a-A) & Fanout2 (S-a-A'of f1) INV2(S-a-A) & LS Wire2 (S-a-A') Fanout2(S-a-A of f1) & LS Wire2 (S-a-A') LS Wire1 (S-a-A')& LS Wire4 (S-a-A')
Full-adder	INV2(S-a-A) & Fanout (S-a-A'of f1) INV1(S-a-A) & LS Wire3 (S-a-A')
2-to-4 Decoder	INV2(S-a-A) & Fanout4 (S-a-A'of f1)

Table 5.21
Fault Coverage of Two Defects (One per Device)

Circuit	F_C of Exhaustive Test Set (%)	Minimal Test Set Cardinality	F_C of Minimal Test Set (%)
EXOR	$111/116 = 95.7$	2	$103/116 = 88.8$
Full adder	$73/75 = 97$	2	$71/75 = 94.6$
2 to 4 Decoder	$143/144 = 99$	3	$143/144 = 99$

Let the dimension of a square cell and the cell-to-cell distance be denoted by l and k, respectively; in this section, constant scaling is investigated, i.e., l/k is constant.

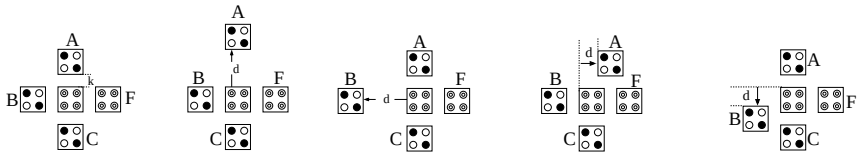
5.2.6.1 MV

As mentioned before, the basic functionality of QCA is based on the Coulombic interaction among neighboring cells; this depends on the distance as well as the angle between cells.

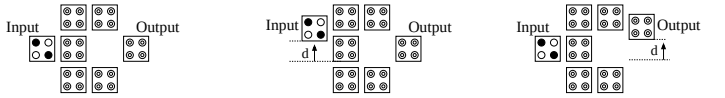
Consider initially, a defect-free MV [Figure 5.24(a)(1)] for $k = l/4$. The simulated waveforms for scaling the MV using both engines are shown in Figure 5.25(a) (note that in each waveform the Y-axis has a different scale). The simulation results confirm that an MV made of small sized cells is rather robust (i.e., the correct output function has a sharp-edged waveform and high polarization level). However, when l increases (up-scaling), the polarization level drops; glitches and distortion appear in the waveform. For a high value of l, the MV ceases to function correctly; for example in Figure 5.25(a), when $l = 70nm$, in addition to glitches and distortion, the output is equal to B (rather than the MV of the inputs). Moreover, the output polarization depends on the input pattern. Some input patterns generate higher polarization levels; the difference in polarization level among input patterns also increases with an increase in cell size.

The two engines generate similar results; however the bistable engine shows less distortion and glitches compared to the coherence vector engine. Simulation has shown that both engines provide the correct logic output for MV with $l < 40nm$. For $40 \leq l < 45nm$, the coherence vector engine shows a substantially higher level of distortion than the bistable engine. When l exceeds $45nm$, the waveforms for both simulator engines show a higher level of distortion and reduced polarization at the output, thus generating an erroneous value at the output. This suggests that

the approximation used in the bistable engine is not always accurate at large cell dimension.



(a) MV displacement and misalignement faults



(b) INV displacement faults

Figure 5.24 Displacement and Misalignment Defects in MV and INV

Consider next scaling in the presence of defects, that is, various displacement and misalignment defects (see Figures 5.24(a)) are introduced in the MV. Let the smallest distance by which a cell is moved for generating an erroneous output be denoted by d. d is referred to as the *scaling defect tolerance*. Using both engines, it has been found that d is the same, as shown in Figures 5.26. The best-fitting curves (in a polynomial of fifth degree) are shown in these figures. For all simulated defects, d first increases with cell size, then it levels off to a constant level; further increase in l results in a decline of d (in many cases, this is quite steep). The results also show that MVs with smaller cells have a value of d/l higher than for MVs made of larger cells (i.e., the scaling-down process improves robustness). Although the same d is obtained with both engines, the output waveform by the coherence vector exhibits more distortion and glitches when the cell size is increased. Figure 5.27(a) shows the results simulated with both engines for an MV with $l = 30$ and a displacement for input cell B of $87nm$.

Misalignment defects have also been simulated; the defect tolerance is $d \leq l/2$, as a shift of half a cell generates logic inversion in QCA. Similar to the displacement defects, the defect tolerance for misalignment of A, C and F shows a saturated shape. However, the defect tolerance for misalignment of B is linear with respect to the cell size l.

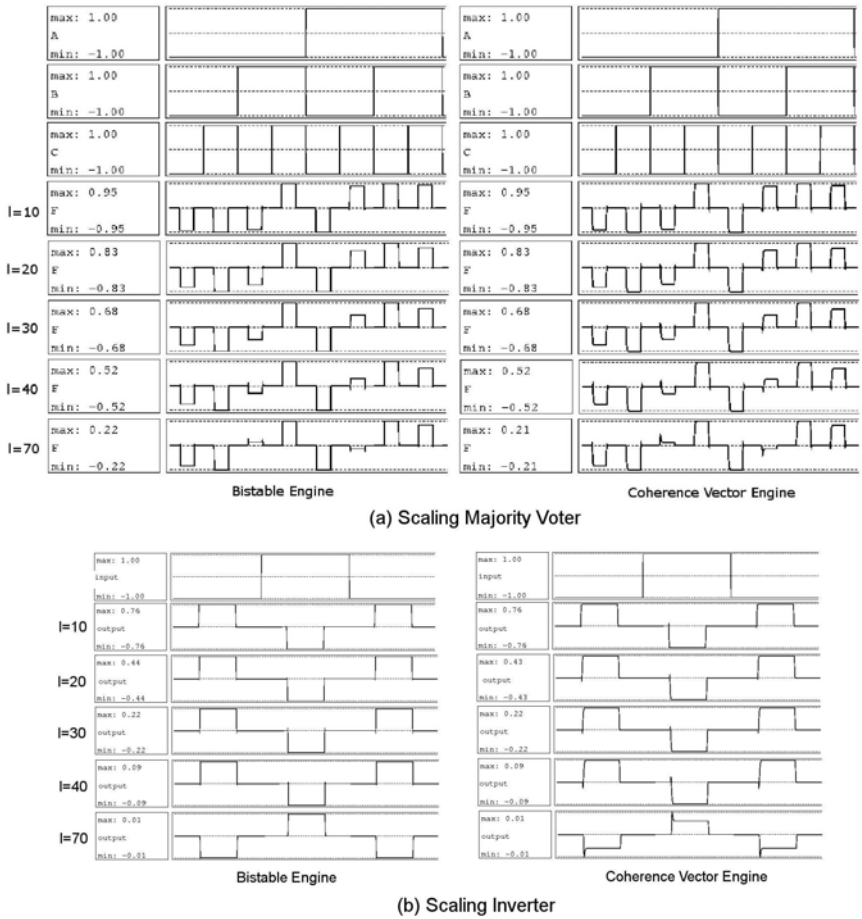


Figure 5.25 Simulation Results for Scaling MV and INV

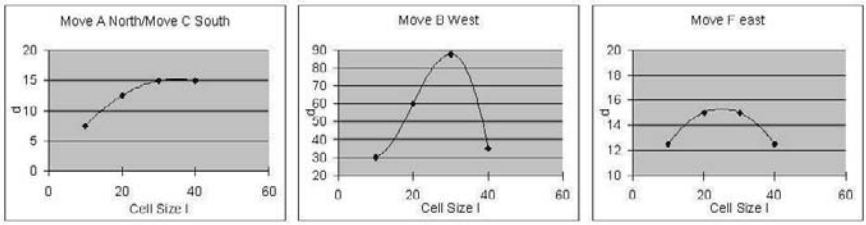


Figure 5.26 Scaling for Displacement Defects in MV

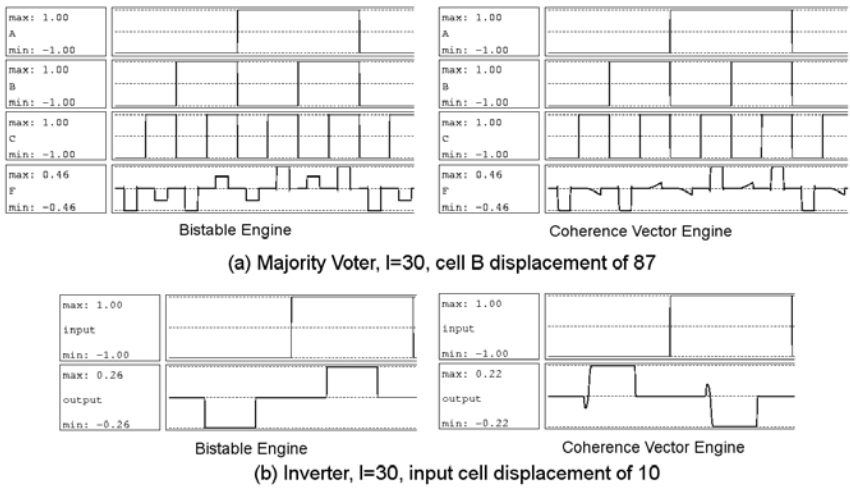


Figure 5.27 Waveforms from the Simulation Engines for Displacement Defects in MV and INV

5.2.6.2 INV

The seven-cell INV (Figure 5.24(b)) is investigated and the simulated waveforms for scaling the INV with both engines are given in Figure 5.25(b). Similar to the results of MV, INVs made of smaller cells are more robust and the coherence vector engine shows more distortion than the bistable engine. Input and output cell displacement defects (Figures 5.24(b)(2) and (3)), have been considered. As for the MV, both engines give the same results for d. However in some cases different waveforms, for example an INV with $l = 30$ and input cell displacement of $10nm$ (Figure 5.27(b)), can be observed.

5.2.7 Conclusion

In this chapter, a detailed simulation-based defect characterization for QCA logic and interconnect devices has been presented. These results are then used to investigate the defect tolerance of QCA reversible systems.

Various failure mechanisms that can potentially happen during nano manufacturing of these devices, have been considered and simulated. These include missing cell, extra cell, displacement and misalignment defects. Simulation results show that the behavior of QCA defects at a logic level (i.e., faults) is not similar to conventional faults in CMOS technology. For example, an *unwanted complementation* fault at logic-level has been observed for a considerable number of cases of coupling defects for both interconnects and active devices. The bridging mechanisms between QCA wires, either in binary wires (interconnects) or inside logic devices (such as those among the wires of the AOI gate) are quite different from conventional wired-or and wired-and bridging faults in a CMOS design. Hence, appropriate fault models for QCA must be developed and used for test generation.

Extensive simulation results have been provided for defect characterization and fault analysis. An interesting effect that has been observed in molecular QCA is the so-called undetermined fault (lack of polarity or presence of glitches in a signal). It has been shown that this type of fault can be propagated and detected only when the output of the affected device is a primary output, that is, when a device that is affected by this type of fault is located internally to the QCA circuit, the undetermined value never propagates and the correct output is always observed at the primary output. This is caused by the regenerative effect of the non linear nature of the cell-to-cell QCA response, as a weak polarization appears only at an isolated output; however, when the output is not isolated, cell-to-cell interaction

causes the regeneration of the weak polarization and therefore, the non propagation of the functional fault occurs.

A defect-driven approach has been proposed for testing molecular QCA circuits. A novel and effective metric for grading has been introduced based on a probabilistic analysis of the layout. The likelihood of occurrence of functional faults under such probabilistic analysis has been derived for many QCA devices. This metric has been utilized as a criterion for selecting and prioritizing vectors when testing molecular QCA circuits. Testing of few QCA circuits (such as EXOR, Full Adder, 2 to 1 Decoder) has been analyzed in detail. This has confirmed that the device-level analysis can be extended to circuit-level, while obtaining consistent results for the functional fault set and coverage. Simulation results in the presence of one and two defective devices per QCA circuit (assuming one defect per device) have been provided and have confirmed the validity of the proposed analysis.

Scaling has also been analyzed to establish the tolerance to different defects, as caused by variations in the QCA manufacturing process. It has been shown that for QCADesigner, the coherence vector engine shows more distortion than the bistable engine. For displacement defects and most misalignment faults, the relationship between cell dimension and defect tolerance shows an increase of defect tolerance with cell size increase and then leveling off to a constant level and rapidly declining as the cell size increases further (in a shape of a parabola). Small cells exhibit strong Coulombic interactions and therefore have better defect tolerance and robustness for scaling.

References

[1] Ravichandran, R., et al., "Automatic Cell Placement for Quantum-dot Cellular Automata," *Great Lake Symposium on VLSI (GLSVLSI)*, 2004, pp. 332-337.

[2] Tahoori, M., and F. Lombardi, "Testing of Quantum Dot Cellular Automata Based Designs," *Design Automation and Test in Europe (DATE) Conference*, 2004, pp. 1408-1409.

[3] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also available online: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

[4] Jiao, J., et al., "Building Blocking for the Molecular Expression of QCA, Isolation and Characterization of a Covalently Bounded Square Array of two Ferrocenium and Two Ferrocene Complexes," *Journal of the Am. Chem. Society (JACS Communications)*, Vol. 125, No. 25, 2003, pp. 7522-7523.

[5] Qi, H., et al., "Molecular Quantum Cellular Automata Cells: Electric Field Driven Switching of a Silicon Surface Bound Array of Vertically Oriented Two-Dot Molecular QCA," *Journal of the Am. Chem. Society, (JACS Articles)*, Vol. 125, No. 49, 2003, pp.15250-15259.

[6] Lent, C. S., B. Isaksen and M. Lieberman, "Molecular Quantum-Dot Cellular Automata," *Journal of the American Chemical Society*,Vol. 125, No.4, 2003, pp. 1056-1063.

[7] Hu, W., et al., "High-Resolution Electron Beam Lithography and DNA Nano-Patterning for Molecular QCA," *IEEE Transactions on Nanotechnology*,Vol. 4, No. 3, 2005, pp. 312-316.

[8] Personal communication with Professor Marya Lieberman, Department of Chemistry and Biochemistry, University of Notre Dame, IN, USA.

[9] Frost, S. E., et al., "Carbon Nanotubes for Quantum-Dot Cellular Automata Clocking," *IEEE Conference on Nanotechnology*, 2004, pp. 171-173.

[10] Armstrong, C. D., W. M. Humphreys, and A. Fijany, "The Design of Fault Tolerant Quantum Dot Cellular Automata Based Logic," *11th NASA Symposium on VLSI Design*, 2003.

[11] Fijany, A., and B. N. Toomarian, "New Design for Quantum Dots Cellular Automata to Obtain Fault Tolerant Logic Gates," *Journal of Nanoparticle Research*, Vol. 3, No. 1, 2001, pp. 27-37.

[12] Fijany, A., N. Toomarian, and K. Modarress, "Block QCA Fault-tolerant Logic Gates," Technical Report, Jet Propulsion Laboratory, California, 2003.

Chapter 6

Two-Dimensional Schemes for Clocking/Timing of QCA Circuits

V. Vankamamidi, M. Ottavi and F. Lombardi

QCA has the advantages of low power dissipation, potential for high throughput due to efficient pipelining, fast signal switching and propagation. Conventional QCA circuits use the one-dimensional clocking scheme, as introduced previously in Chapter 3. However with this clocking scheme, QCA designs of even modest complexity suffer from the negative impact due to the placement of long lines of cells among clocking zones, thus resulting in increased delay, slow timing and sensitivity to thermal fluctuations.

In this chapter, we consider issues pertaining to timing and clocking of QCA systems for high performance computing and kink-free (error-free) behavior. Initially, we study the effects of thermal fluctuations on QCA designs as a function of their size. It will be shown that tolerance to thermal fluctuations and high performance computing necessitate a different mechanism than the one-dimensional criteria of clocking proposed in [1]. To address this problem, a novel strategy is proposed for timing and clocking of QCA systems. This strategy is based on a two-dimensional characterization of information transfer across different timing zones arranged into grids. Issues such as clocking circuitry (as interfaced to CMOS) and operating temperature, are also addressed. Novel logic propagation techniques are also introduced for designs under the proposed clocking schemes. Computational time and pipelining are extensively analyzed as some of the performance metrics. The proposed clocking schemes utilize the equivalence between systolic processing and QCA zone switching, thus permitting sequential or parallel timing processing

of signals across both dimensions of the QCA circuit in a Cartesian plane. Simulation results (using QCADesigner [2]) are provided for combinational and sequential QCA circuits.

6.1 CLOCKING ANALYSIS

For QCA, *adiabatic switching* is commonly preferred to abrupt switching [1]. The four phase adiabatic switching scheme was introduced in Chapter 3. In an adiabatic approach, switching is accomplished by modulating the interdot tunneling barrier of the QCA cells. By applying an input signal, barriers are lowered such that cells begin to polarize. By raising back the barriers, cells are held or "crystallized" in their new states. If the change in the interdot potential barrier is gradual, then adiabatic theory guarantees that the system always remains in the *ground state* and does not permanently move to excited or metastable states [3]. A system is said to be in the *ground state* if it has minimum energy, i.e., all cells polarize and attain a state as expected by cell-to-cell interactions. In an excited state, cells align contrary to cell-to-cell electron repulsion and a *kink* is said to have occurred.

In an adiabatic switching scheme, fluctuations in operating temperature may excite QCA cells above their ground state and produce erroneous results at the output. An analysis of these thermal effects on a linear array (or line) of QCA cells is provided [4] . Let E_k represent the energy required for a QCA cell to encounter kink (i.e., to align differently from its expected polarization). As the number of QCA cells in the linear array increases, then the ground state remains unique and the energy separation between the ground state and the first excited state remains E_k. However, with an increasing number of cells, the number of locations increases and so multiple kinks may occur. Therefore, the probability for kink-free behavior is a function of N (as denoting the number of cells in the array). Also at nonzero Kelvin, the higher the operating temperature (T), the higher are the thermal fluctuations that may lead to an increase in the probability of kink occurrence. Finally, the probability for a system to be in an excited state (kink) is a function of the energy required for a kink to occur in a QCA cell, E_k. A higher value of E_k reduces the probability of kink occurrence (with scaling of cell dimension to a molecular-level the correlation between electrons in neighboring cells increases, thus resulting in an increase of E_k). For N QCA cells, these parameters are quantified in the following equation (derived in [4]):

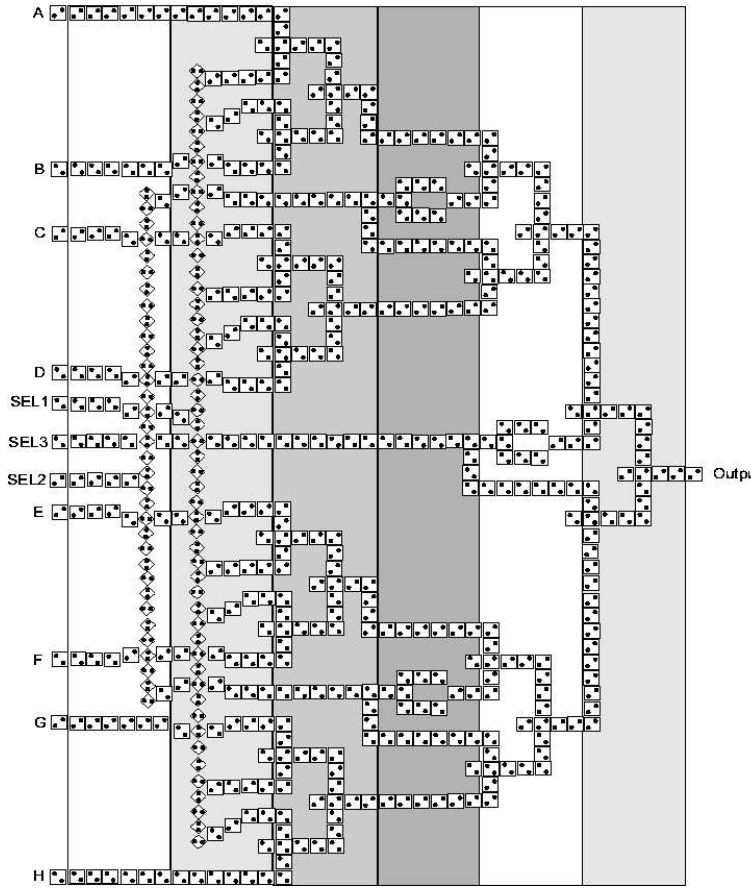


Figure 6.1 8-to-1 QCA multiplexer, One-dimensional Clocking

$$\Delta F_n = nE_k\left[1 - \frac{k_B T}{E_k}\ln(N)\right] \tag{6.1}$$

ΔF_n is the energy separation between the ground state and the nth excited state (i.e., a zone with n kinks and k_B is Boltzmann's constant). As long as the energy separation ΔF_n is greater than zero, then the QCA system does not settle in an excited thermodynamic equilibrium state. This implies that the energy required for *n* kinks (nE_k) must be greater than for the kinks caused by thermal fluctuations, $K_B T\ nln(N)$. From this inequality, for a given kink energy E_k and operating temperature T, a bound on the number of QCA cells for avoiding kinks is given by

$$N \leq e^{\frac{E_k}{k_B T}} \tag{6.2}$$

The bound on line (array) length obtained from (6.2) can be utilized in determining the largest zone dimension under the worst case conditions. Consider a bound on N for the vertical and horizontal dimensions of a zone. From (6.2), thermodynamic effects can then be avoided in all QCA lines within that zone. Therefore, kink-free behavior can be accomplished by establishing an upper bound on N for the dimension of a clocking zone. For QCA pipelining, only one zone (among a set of four adjacent zones) is in the Switch phase at any time; so, the effective length of a long QCA line (that may span across multiple zones) must be equal to the dimension of the switching zone.

6.2 TWO-DIMENSIONAL QCA CLOCKING

The QCA clocking mechanism proposed in [1] partitions a design into different zones only along one direction of signal flow (i.e., the X-axis). Such a scheme considers long horizontal lines and divides them among multiple (vertical) clocking zones, thus keeping their length bounded in any zone. A vertical line (in the Y-axis) is always contained within a column as a single clocking zone; for complex designs, the height of a clocking zone (along the Y-axis) could be significant, thus creating long vertical lines.

Consider the QCA design of 8-to-1 multiplexer as shown in Figure 6.1; throughout this chapter, this is used as a representative circuit for comparison purposes among the proposed clocking schemes. This circuit is designed using three ($log_2(8)$) stages of 2-to-1 multiplexers. The four 2-to-1 multiplexers in stage 1 reduce the eight inputs to four based on the select signal SEL1. Two 2-to-1

multiplexers in stage 2 reduce these four signals to two based on SEL2 and finally a 2-to-1 multiplexer in stage 3 selects one of its two inputs as output based on SEL3. Each 2-to-1 multiplexer is designed using three majority voters (two as AND gates and one as OR gate) and an inverter. As the SEL1 signal must be supplied to all 2-to-1 multiplexers in stage 1, a long vertical line is required (51 cells long in clocking zone 2). The length of the vertical line increases with multiplexer size (N) because the select signal must be supplied to $N/2$ 2-to-1 multiplexers in stage 1.

The problem of long vertical lines is solved in this chapter by partitioning the QCA design along the Y-axis (row-wise) in addition to the X-axis (column-wise). This two-dimensional (2D) arrangement effectively generates a *grid of clocking zones* for a given QCA design. A bound for the zone dimensions restricts the length of the QCA lines and makes QCA designs tolerant to thermodynamic effects. Designs of QCA systems are characterized by the so-called "tournament bracket" structure [5]. Logic signals propagate horizontally through the majority voters by providing outputs toward the end of the bracket. This feature favors partitioning of designs into multiple clocking zones along the X-axis, i.e., horizontal propagation is accomplished. By having a clocking mechanism for two-dimensional partitioning (as for a grid of zones), extensive modifications to the original QCA design must be avoided (if possible). Similarity must be retained in signal propagation such that all zones in a column of the two-dimensional grid must be switched (prior to switching zones located in the next column). Figure 6.2 shows signal propagation for the proposed two-dimensional partitioning of QCA designs. Signals propagate vertically in each column; after switching all zones in a column, the signal propagates horizontally to the next column of the grid. Therefore at a reduced frequency (which is proportional to the number of zones in a column), signal propagation along the X-axis is still equivalent to the one-dimensional clocking case.

For correct operation of the QCA design, all signals in a clocking zone must be made available to the next stage during its Switch phase. In the two-dimensional case, a signal must propagate both vertically and horizontally. So, if a zone in the Hold state is released as soon as the next zone in the same column completes the Switch phase, then its signals will not be available during the Switch phase of the corresponding zone in the next column. This inhibits signal propagation along the X-axis, leading to a possible incorrect behavior of QCA systems. So, all zones in a column must be retained in the Hold state until the corresponding zones in the next column are in the Switch phase. In the clocking mechanism for QCA, a zone is released as soon as the next zone is switched. The proposed mechanism for two-dimensional signal propagation in a grid is similar to the one-dimensional case because a zone can be released as soon as the zones located next (along both

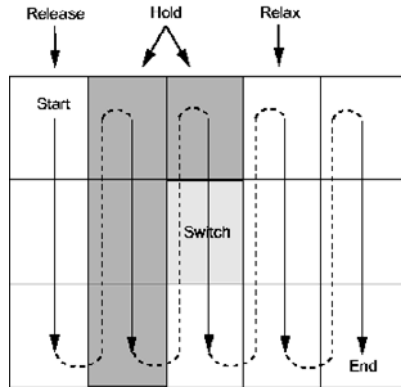


Figure 6.2 Proposed Two-dimensional QCA Clocking

dimensions) are switched. Similarly, a zone can be switched only when its driving zones (in both dimensions) are in the Hold state.

The proposed two-dimensional clocking mechanism requires changes (albeit minor) to existing QCA designs (based on one-dimensional clocking). Changes are required to preserve the direction of logic propagation in QCA lines as shown in Figure 6.2. Clocking requirements and changes in design are summarized by the following rules:

1. Switch all zones in a column prior to switching the zones in the next column.
2. Keep an entire column in the Hold state until all zones located in the next column are switched.
3. Vertical lines spanning multiple zones should accept signals only in the zone from which they originate (this is referred to as *Design Modification-I, DMI*)
4. Signals should not travel in a direction opposite to logic propagation, neither within a column nor between columns (this is referred to as *Design Modification-II, DMII*).

Figures 6.1 and 6.3 show the QCA design of the 8-to-1 multiplexer under the original one-dimensional and the proposed two-dimensional schemes. The design modification rules given previously have been applied to this circuit (DMI is applied to SEL1 and SEL2, $DMII$ is used for the majority voter in the last zone of the grid); using the logic propagation shown in Figure 6.2, the clock zone

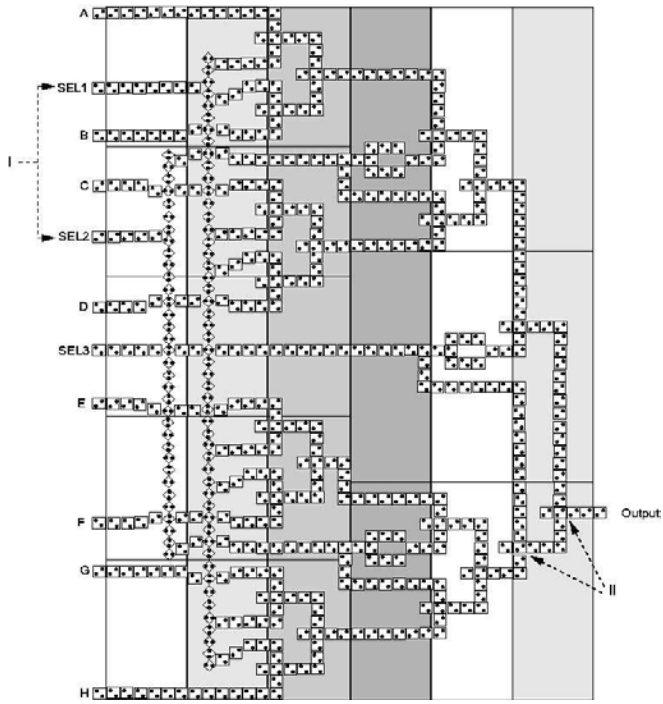


Figure 6.3 8-to-1 QCA Multiplexer, Two-dimensional Clocking.

dimensions in Figure 6.3 are in the order of tens of cells along both axes. This is consistent with clocking zone widths suggested by other works [1] [5] with partitions along one dimension only. Note that vertical lines receive signals in the zone from which they originate and a majority voter has been moved down within a column to avoid inter-zone signal transfer in a direction opposite to logic propagation. As shown in the multiplexer of Figures 6.1 and 6.3, the circuits are almost the same and therefore they occupy the same area. As for the count of QCA cells, the design modifications introduce an overhead that is negligible, i.e., the number of QCA cells in Figure 6.1 is 562, whereas in Figure 6.3 it is 576 for a 2.5% increase.

As a signal in a QCA line propagates through the sequential switching of cells from the input to the output, intuitively, it would take twice as long to switch twice as many cells in a QCA line. The relationship between switching time and number

of cells for a given error margin (as related to non adiabatic ringing) can be assessed by solving the time-dependent Schrödinger equations. The solution is provided in [1] by giving the dependence of minimum switching time on the number of cells in a line as,

$$T_s \propto C^{1.16} \tag{6.3}$$

where T_s is the minimum switching time and C is the number of cells in a line. The exponential factor of 1.16 suggests that switching time has almost a linear dependence on the number of cells ($O(C)$). The small deviation from linearity is the result of fitting the maxima for error (non-adiabatic ringing) in solving Schrödinger equations.

The minimum clock period for a clocking zone is determined by the switching time of the longest QCA line in that zone. In most cases, the length of the longest QCA line is proportional to the vertical and horizontal dimensions of a zone. Therefore, even though the number of zones per column in a grid is increased, the minimum clock period for each zone is reduced due to smaller zone dimensions. So, there is a linear relationship between the clock period of a column with no partition (as in the original one-dimensional scheme) and a column with partitions (as in the proposed scheme).

The total computation period is the sum of the clock periods of all columns in the QCA design; this is almost the same for both the one-dimensional and two-dimensional schemes. Pipelining is not affected because an entire column is used to hold the signals. In both clocking schemes, four columns are required to propagate one state of computation. For the 8-to-1 QCA multiplexer, the proposed two-dimensional scheme reduces the longest vertical line length from 51 cells to 13 cells (as shown in column 2 of Figures 6.1 and 6.3). From Equation 6.2, for a line of 51 cells to avoid kinks, the excitation energy E_k of the cells must be 3.9 times greater than k_BT; for a line of 13 cells it only needs to be 2.6 times greater. So, for a given QCA technology (i.e., for a fixed E_k), if the 8-to-1 QCA multiplexer using the proposed two-dimensional clocking scheme can be operated at room temperature (300 degrees Kelvin), then the one-dimensional version of the same circuits must be operated at 195 degrees Kelvin. However the CMOS clocking circuit which is required for the two-dimensional scheme is more complicated than the one-dimensional scheme. A detailed discussion of this topic is provided in a later section of this chapter.

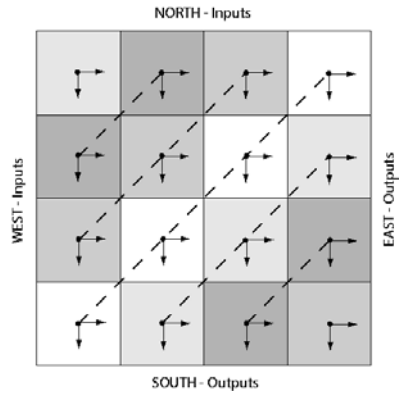


Figure 6.4 Clocking for Two-dimensional Wave Propagation

6.3 TWO-DIMENSIONAL WAVE QCA CLOCKING

Significant improvement in computation time and simplification of clocking circuitry can be achieved by employing a different clocking mechanism for QCA designs partitioned along two-dimensions. This new scheme is based on the *parallel* execution and processing in the clocking zones within a different timing framework.

The principles of this technique are based on the similarity between systolic arrays and QCA with respect to clocking. Systolic arrays are special purpose VLSI architectures introduced in the late 1970s [6]; they are made of simple processing elements with local interconnections usually arranged in a grid layout. Each processing element receives data from one or more neighboring processing elements (at its primary inputs); it then performs local computation and transfers its results to other neighboring processors (connected to its primary outputs). Two-dimensional (square) systolic arrays used for parallel processing of matrix multiplication accept inputs from two sides of a square and propagate the outputs to two other sides. As a partitioning scheme for clocking zones, the proposed two-dimensional arrangement is similar to a grid with orthogonal interconnections. Computational results move from north-west to south-east, this is similar to the implementation in a two-dimensional (square) systolic array. Due to these similarities, logic-wavefront propagation techniques developed for systolic arrays can be considered for QCA architectures to increase data pipelining and parallel processing [7].

Figure 6.4 shows a logic propagation technique for the proposed two-dimensional diagonal wave scheme (2DDWave). To retain similarity to the two-dimensional (square) systolic array (and thereby achieve parallel processing), each zone must accept input signals only from two zones (north, west) and pass its outputs to the other two zones (south, east), that is, each column should have an *equal number* of zones (perfect grid). Therefore, to ensure an efficient utilization of the wavefront propagation scheme, a design modification rule must be applied in addition to the rules presented for the two-dimensional QCA clocking scheme of the previous section, i.e., the design must be partitioned into a *perfect grid* of zones such that, all zones in a row have the same height and all zones in a column have the same width. Figures 6.3 and 6.5 show the 8-to-1 QCA multiplexer before and after the above design modification rule. In this *perfect grid* scheme the correct switching of a zone requires only two zones (one located above the Switch phase zone and one located to the left of the Switch phase zone) to be in the Hold phase. Similarly, a zone needs to be in the Hold state only until the zones located below (south) and right (east) are switched. With this switching arrangement, the proposed diagonal wavefront propagation scheme (denoted as 2DDWave) produces at the output the same results as the one-dimensional and two-dimensional schemes presented previously.

In a one-dimensional clocking scheme, the lengths of the vertical lines are not bounded because they increase as a function of design size. As the operating temperature (T) changes with the number of cells (N) in the longest QCA line of a clocking zone, then T becomes a function of design size. However, in the proposed two-dimensional schemes, independent of the design size, line lengths can be bounded as partitioning occurs along both the X-axis and Y-axis. Therefore, QCA designs under two-dimensional schemes are robust to thermal fluctuations and can be operated at higher temperatures, mostly independent of size. In a two-dimensional scheme (2D), the underlying feature is the *sequential processing* in a linear fashion. All zones in a column are switched sequentially, prior to switching zones in the next column (Figure 6.2). In the proposed two-dimensional wave clocking scheme (2DDWave), switching is performed in *parallel*; all zones that are located *along the diagonals* are *switched* simultaneously. Therefore, the computation time for the 2D scheme increases quadratically with the number of zones along the X-axis and Y-axis (given by $Z_x \times Z_y$), whereas in the 2DDWave scheme, the increase is linear ($Z_x + Z_y$). However in a previous section it has been shown that the computation times for the 1D and 2D schemes are equivalent. Therefore, the proposed 2DDWave scheme performs better in terms of processing speed than these two schemes.

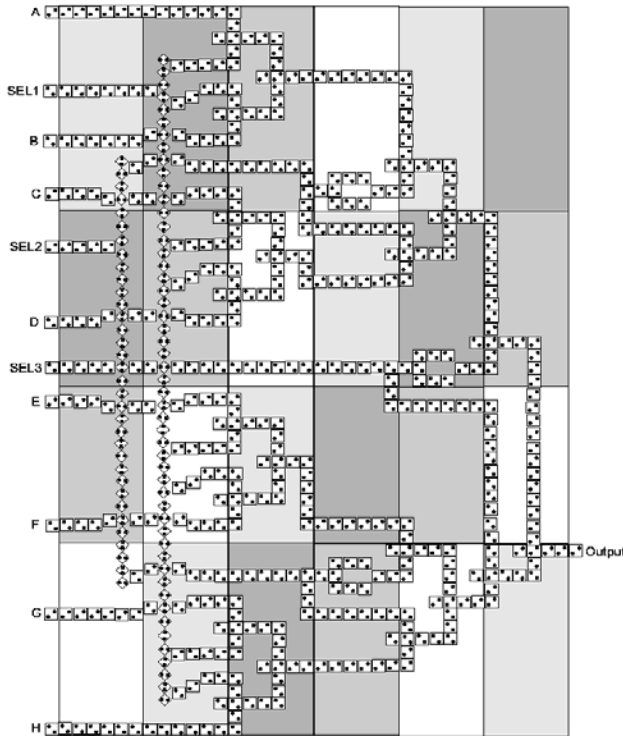


Figure 6.5 8-to-1 QCA Multiplexer, Two-dimensional Wave Clocking

Table 6.1
Comparison of Different Clocking Schemes.

$Characteristics$	$1D$	$2D$	$2DDWave$
# of Cells (C)	575	576	576
# of Zones (Z)	6	24	24
Max. Wire Len (L)	51	13	13
$\frac{E_k}{kT}$ for kink-free operation	3.9	2.6	2.6
Max. Temp for kink-free operation	195k	300k	300k
Computation Time	$\sim$24 time units	$\sim$ 24 time units	$\sim$ 9 time units
Pipelining	Four-staged	Four-staged	Four-staged
Clocking Circuitry	Modest	Complex	Modest

Table 6.1 shows the characteristics of the three clocking schemes discussed in this chapter for the 8-to-1 QCA multiplexer design (Figures 6.1, 6.3 and 6.5) as example. As the underlying feature of both two-dimensional schemes is to partition the QCA system along the X and Y axes, they have common characteristics of kink resilient behavior and a higher operating temperature (as discussed previously). However as an additional advantage, the 2DDWave scheme improves computation time.

As reported previously [8] [1], QCA designs can be clocked by an electric field generated by a set of parallel CMOS wires buried under the substrate. For the one-dimensional scheme, these metal wires are vertically oriented such that columns of clocking zones are formed. By keeping the set of four adjacent metal wires out of phase by $\frac{\pi}{2}$ and applying the signal shown in Figure 6.6, clocking requirements can be satisfied. However, clocking in the two-dimensional (2D) case is more complicated because all zones in a column are clocked simultaneously during the Hold, Release and Relax phases, but they are clocked sequentially during the Switch phase. Therefore to provide phase-based clocking a CMOS circuitry must supply multiple signals; moreover, multiplexing between them is also required (the reader should refer to [9] for additional details).

The two-dimensional wave (2DDWave) scheme requires a simpler arrangement because all zones along the diagonals are clocked simultaneously in all phases. However in this case, the set of parallel metal wires runs diagonally to the QCA design, i.e., a wire runs under all clocking zones located diagonally to each other. To

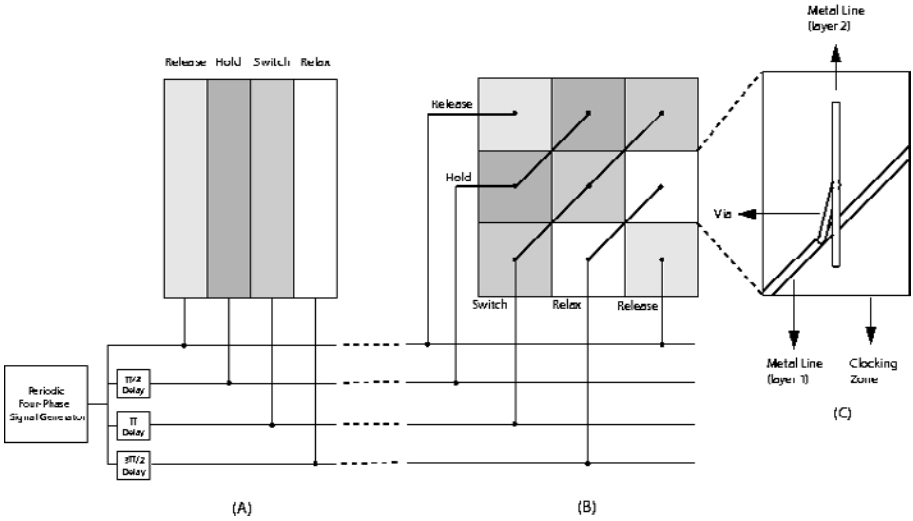


Figure 6.6 CMOS Clocking Circuitry for QCA Designs. A) Circuitry for One-dimensional (1D) clocking scheme. B) Clocking Scheme for Two-dimensional Wave (2DWave) Clocking Scheme. C) Second Layer of Metal Wires to Provide Uniform Electric Field Over a Clocking Zone in 2DDWave Scheme.

provide an uniform electric field across a clocking zone two layers of metal wires are required as shown in Figure 6.6. The diagonal metal wires run in layer 1 (bottom) over the entire QCA design; metal wires in layer 2 (top) are small, disjointed and extend only over a single clocking zone to provide a uniform electric field. Metal wires in layer 1 and layer 2 are insulated through an oxide layer such that the electric field generated by metal layer 1 does not interfere with the electric field of metal layer 2. The signal in metal layer 1 is transferred to the metal wires in layer 2 (for the diagonal clocking zones) through vias. A ground plane (not shown in the figure) can be added on top of the QCA layer to reduce fringing effects for the lines of the E field.

Logic-level effects due to interference in the electric field between adjacent metal wires used for clocking are minor because cells that are at the boundary must belong to either of the two adjacent clocking zones (depending on the strength of the electric fields in the corresponding layer 2 metal wires). So the interference of electric fields can be tolerated by designing circuits such that QCA cells at clock

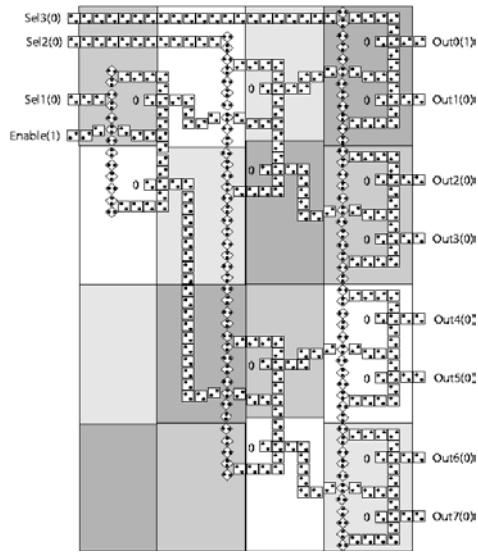


Figure 6.7 3-to-8 Decoder Under 2DDWave Clocking Scheme

zone boundaries can belong to either of the clocking zones and still not modify the logic functionality.

The 8-to-1 multiplexer design can be extended to other circuits with similar functionality. Figure 6.7 shows the QCA design of a 3-to-8 decoder under the 2DDWave scheme. This circuit can be used in interconnection networks and for memory address decoding [10]. The design of this circuit is similar to the 8-to-1 multiplexer (shown in Figure 6.5); it uses few majority voters reduced to AND/OR gates at each of the $log_2(n)$=3 stages to decode the address. Figure 6.7 illustrates the design modifications that are required under the 2DDWave scheme to overcome the tournament bracket (tree) structure of the one-dimensional clocking technique.

6.4 EXAMPLES OF QCA CIRCUITS

The 8-to-1 multiplexer was used throughout the previous sections of this chapter as an example circuit. Any combinational circuit whose logic propagation can be confined to the two directions of a 2D plane, can also employ the proposed clocking schemes. In this section, four additional QCA circuits (a full adder, parity

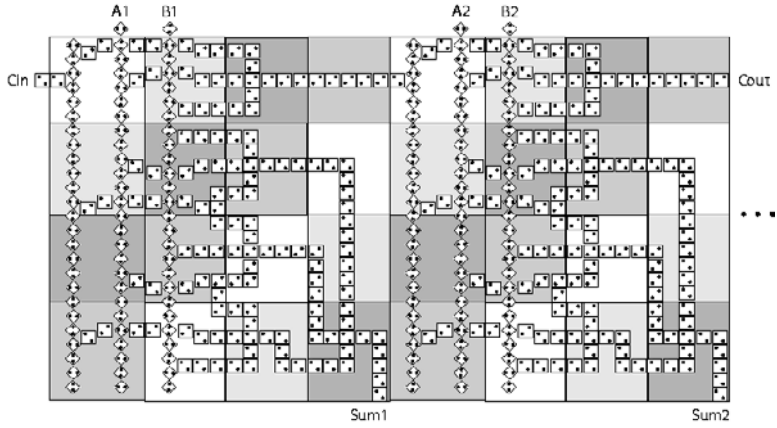


Figure 6.8 Two 1-bit Full Adder Blocks Under 2DDWave Clocking Scheme

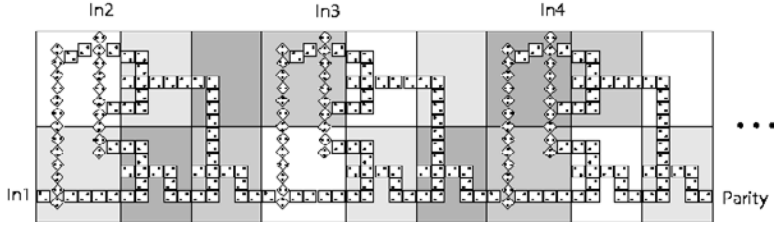


Figure 6.9 Four 1-bit Block Parity Checker Under 2DDWave Clocking Scheme

Table 6.2
Comparison of QCA Designs Under 1D and 2DDWave Clocking Schemes.

	Ripple Carry Adder		Parity Checker		RS FF		3-8 decoder	
Characteristics	1D	2DD Wave	1D	2DD Wave	1D	2DD Wave	1D	2DD Wave
# of Zones (Z)	8	32	9	18	6	18	4	16
Max. Wire Len (L)	25	6	13	7	14	7	48	12
$\frac{E_k}{kT}$ for kink-free operation	3.2	1.8	2.6	2	2.6	1.9	3.9	2.5
Max. Temp for kink-free operation	169k	300k	230k	300k	219k	300k	192k	300k
Computation Time (time units)	∼32	∼ 11	∼ 18	∼ 10	∼ 18	∼ 8	∼ 16	∼ 7

checker, a 3-8 decoder, and the combinational part (no feedback) of a RS Flip-Flop) are presented for the 2DDWave clocking scheme. The first two circuits are also designed as Iterative Logic Arrays (ILA) to allow modularity, improve area efficiency and reduce testing complexity.

Figure 6.8 shows two iterative logic blocks connected together, each realizing a one-bit full adder. The *Sum* output is realized using four majority functions as $Sum = Maj(Maj(A, B', C_i), Maj(A, B, C_i'), Maj(A', B, C_i))$. The carry out ($C_{out}$) is realized using only a single majority voter as $C_{out} = AB + BC_i + C_iA$. All inter-zone signals propagate only along the two dimensions (horizontal: left-to-right, vertical: top-to-bottom), thus satisfying the requirement for the 2DDWave clocking scheme. By concatenating n of these blocks into an ILA, a n-bit ripple carry adder can be constructed. Figure 6.9 shows the QCA design of a four bit parity checker implemented using three blocks of iterative logic. Each block is a two input XOR gate designed using two AND gates and an OR gate. Signal complementation is accomplished by vertical wires that act as inverter chains. Finally two more circuits are considered: the 3-8 decoder shown in Figure 6.7 and the combinational circuit of a RS Flip Flop, shown in Figure 6.15 (this circuit will be described in detail in the next section to introduce the feedback loops). Table 6.2 summarizes the characteristics of these four circuits under the 1D and 2DDWave schemes. Again the improvements in operating temperature and computation time are evident for all considered circuits.

6.5 FEEDBACK PATHS

One of the main issues arising in clocking schemes for QCA is the ability to handle feedback paths. In both a traditional one-dimensional clocking scheme and the proposed two-dimensional clocking schemes, signal propagation is strictly uni-directional (west to east in the 1D case (Figure 6.1) and north-west to south-east in the 2D case (Figure 6.4)). Hence, although clocking schemes are readily applicable to combinational circuits, feedback paths (as in sequential circuits) may require a different technique.

The authors of [5] have proposed a trapezoid clocking mechanism for the one-dimensional scheme to enable feedback paths in QCA designs and better utilize the layout area (by exploiting the tournament bracket structure of QCA circuits). The main principle of the trapezoid approach for handling feedback paths consists of having a sequence of clocking zones to loop backwards along the (feedback) path. This allows a QCA wire in a loop of clocking zones to route a feedback signal even though signal propagation between clocking zones is still uni-directional. The so-called trapezoid mechanism [5] can also be adopted for the proposed two-dimensional clocking schemes to allow feedback paths. Figure 6.10 shows the loop of clocking zones for implementation under a two-dimensional scheme. To allow feedback paths, zones in each region are clocked using the 2DD wave scheme, such that signal propagations are as follows: from north-west to south-east in region 1 and region 2, north-east to south-west in region 3 and region 4, south-east to north-west in region 5 and south-west to north-east in region 6. Thus, circuits in all six regions can receive their outputs as one of their inputs using feedback paths. Circuits can also receive new inputs and propagate their outputs; for example, while region 2 receives a feedback input from west and propagates the feedback path through south, it can receive new inputs from north and send out the outputs through east. It should be noted that if each region in Figure 6.10 has only one zone, then the feedback path reduces to the basic trapezoid clocking mechanism of [5]. A difference in directions of signal propagation in the regions does not result in an added complexity for the underlying clocking circuitry. This occurs because zones in each region are still clocked using the same quasi-adiabatic switching mechanism (consisting of four clock phases) as originated from the wires generating the E field as clock signal. To achieve the required directions of signal propagation, the clock phases of the zones must be scheduled such that switching of the final zone in the 2DD wave of a region is followed by the switching of the first zone in the 2DD wave of the next region, i.e., synchronization of clock phases between regions must be maintained.

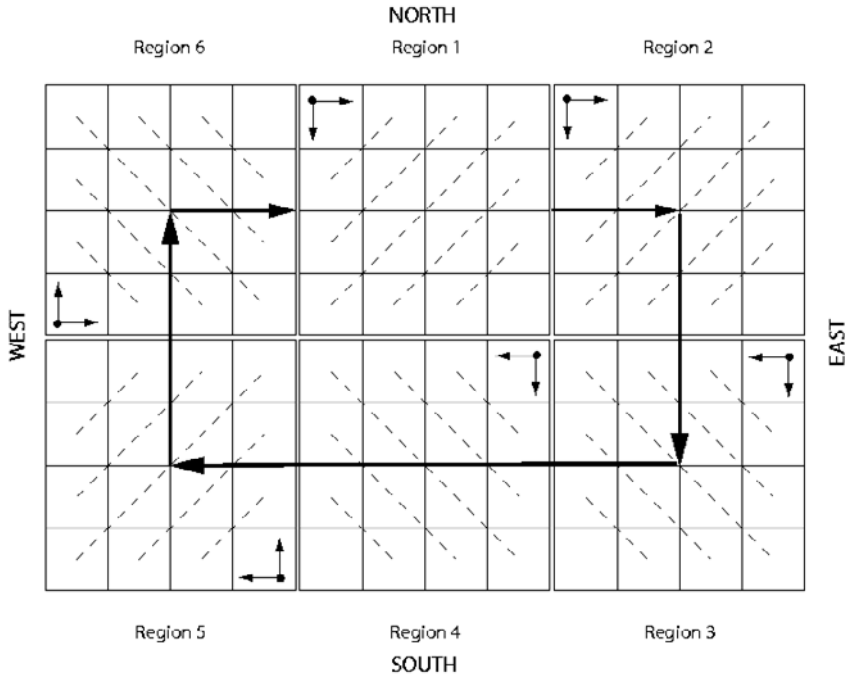


Figure 6.10 Feedback Path for Two-dimensional Clocking Schemes

Thus, the proposed two-dimensional clocking schemes can be used for clocking both combinational and sequential circuits in QCA, while avoiding the problem of kinks and improving performance. The proposed schemes are general; for memories [11], [10] has proposed architectures that also target the problem of kinks by making QCA line length in a clocking zone independent of memory size.

6.6 SIMULATION RESULTS

In this section, the simulation results of the proposed two-dimensional diagonal (2DD) clocking scheme are presented using QCADesigner. Three logic circuits have been designed and simulated by using the 2DD clocking scheme. For all simulations, the QCA cell dimension of 18 nm and dot size of 5 nm are used. Results are obtained using the coherence vector engine of QCADesigner.

Figures 6.11 and 6.13 shows circuits designed in QCADesigner and clocked such that when a two dimensional grid is imposed, all zones along a diagonal are in the same clock phase.

6.6.1 2-to-1 Multiplexer

Figure 6.11 shows the design of a 2-to-1 Mux; it requires two AND gates followed by an OR gate. The 2-to-1 Mux is the building block for larger multiplexers (e.g. a n-to-1 multiplexer built using 2 (n/2)-to-1 multiplexers and a 2-to-1 Mux). The 2D grid is imposed to show that all diagonal zones are in the same clock phase. All design requirements for 2D wave clocking are met as signals flow from north-west to south-east. One delay element is added to the *input Sel* that is at a distance of one clock zone from the top-left zone (as the first zone to be clocked) and four delay elements are added to the *input B*. Each delay element adds a delay of one clock phase. Figure 6.12 shows the result of the simulations for the input and output waveforms. *Input Sel* is defined by the bit string 0000111100001111, *Input B* is given by 0011001100110011, *Input A* is given by 0101010101010101 and therefore *Output Out* is given by $XX00110101001101$ which is the logic behavior of a 2-to-1 Mux. There is a delay of three clock periods because it takes three clock periods for the inputs to reach the output.

6.6.2 One-bit Full Adder

Figure 6.13 shows a one-bit full adder designed using QCADesigner. The implementation of the Carry-Out is not shown as it can be obtained in QCA by using a single MV gate. All the rules and techniques followed in the design of 2-to-1 Mux are also used in this circuit although it is much larger (it requires 9×8 zones). Figure 6.14 shows the simulation results. *Input A* is 00001111, the *Input B* is 00110011, *Input Cin* is given by 01010101 and the *Output Sum* is given $XXXX0110$ which is the logic behavior of a one-bit full adder. There is a delay of five clock periods because it takes five clock periods for the inputs to reach the output.

6.6.3 RS Flip-flop

The proposed 2DD clocking scheme has also been evaluated for a sequential circuit with feedback loop. Figure 6.15 shows a RS flip-flop (proposed in [12]); Figure 6.16 shows the corresponding schematic diagram. The logic part of RS flip-flop is clocked using 2DD clocking such that signal propagation between clocking zones

is from north-west to south-east, whereas the feedback path is clocked such that signal propagation is north-east to south-west (i.e., the feedback path has only one row of clocking zones and signal propagation is from east to west). Delay elements are added to the two inputs R and S such that circuit is balanced, i.e., the number of clock cycles required for the inputs R and S to reach the output Q and the number of clock cycles required for the feedback path are matched. One of the rules for correct behavior of sequential circuits in QCA is that the inputs need to be active for as many clock cycles as required by the feedback path to loop back the output. In the circuit design shown in Figure 6.15, the feedback path requires three clock cycles to loop back the output, i.e., Q(t+1) is obtained after three cycles. So, the inputs need to be active for three cycles. Figure 6.17 shows the result of the simulations for *Input R* as 00011100000000011, *Input S* as 11100000011100000 and *Output Q* as $XX111000000111111$ where X indicates a don't care condition. There is a delay of three clock cycles as it takes three clock cycles for the inputs to reach the output.

6.7 CONCLUSION

With the conventional one-dimensional clocking scheme, QCA designs of even modest complexity suffer from the disadvantage of long vertical lines in the placement of the cells, thus resulting in long delay, slow timing, the inability to operate at higher (room) temperature and sensitivity to thermal fluctuations. In this chapter, we have considered issues pertaining to timing and clocking of QCA systems for high performance computing. Different schemes for clocking and timing of QCA systems have been proposed; these schemes utilize novel two-dimensional techniques that permit a reduction in the longest line length in each clocking zone. In contrast with previous works, the QCA design is partitioned into a grid of zones along both directions (vertically and horizontally) of signal flow. Similar to [1], the proposed arrangements result from the four phases required for correctly operating the QCA cells. These schemes are based on a two-dimensional characterization of information transfer across different timing zones arranged into grids by utilizing logic propagation techniques and modifying cell placement, thus ensuring correct signal generation and timing. The proposed clocking schemes are based on the equivalence between systolic processing and QCA zone switching, thus permitting sequential or parallel timing processing of signals across both dimensions of the QCA circuit.

As novel logic propagation techniques are introduced, computational time and pipelining have been extensively analyzed as some of the most important

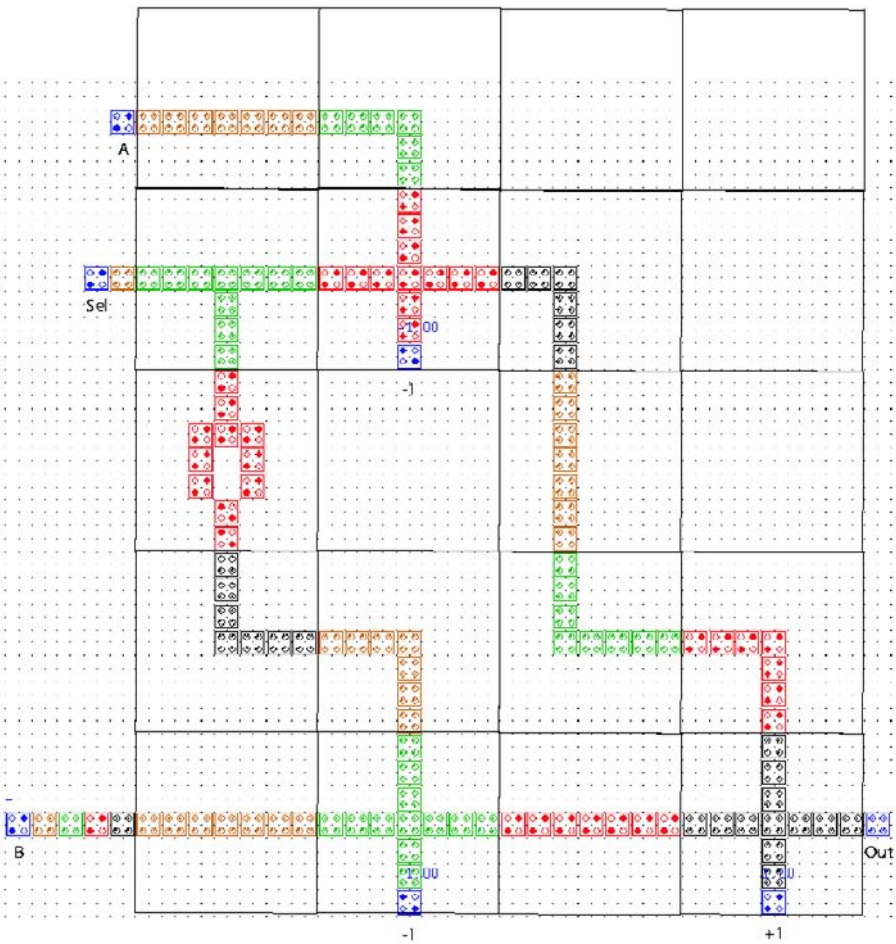


Figure 6.11 2-to-1 Multiplexer Under 2DD Wave Clocking Scheme for Simulation Using QCADesigner

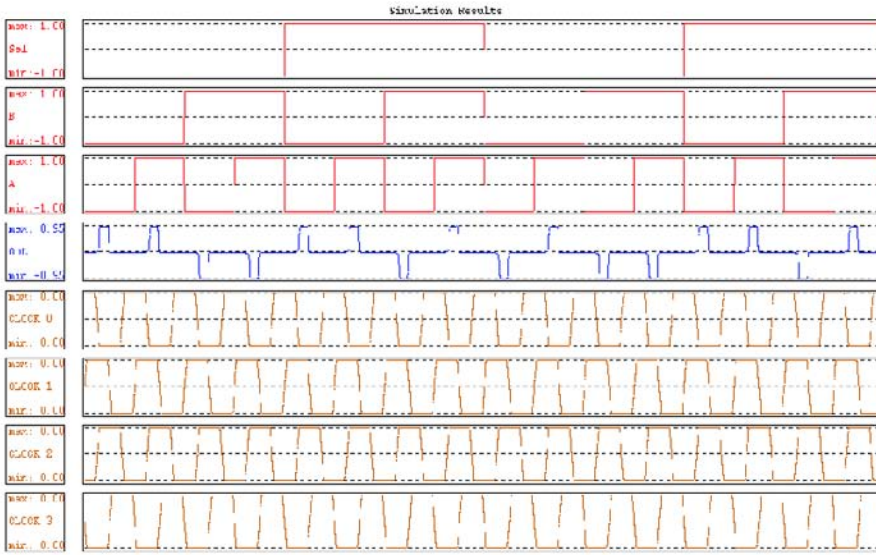


Figure 6.12 Simulation Waveforms for 2-to-1 Multiplexer Under 2DD Clocking Scheme

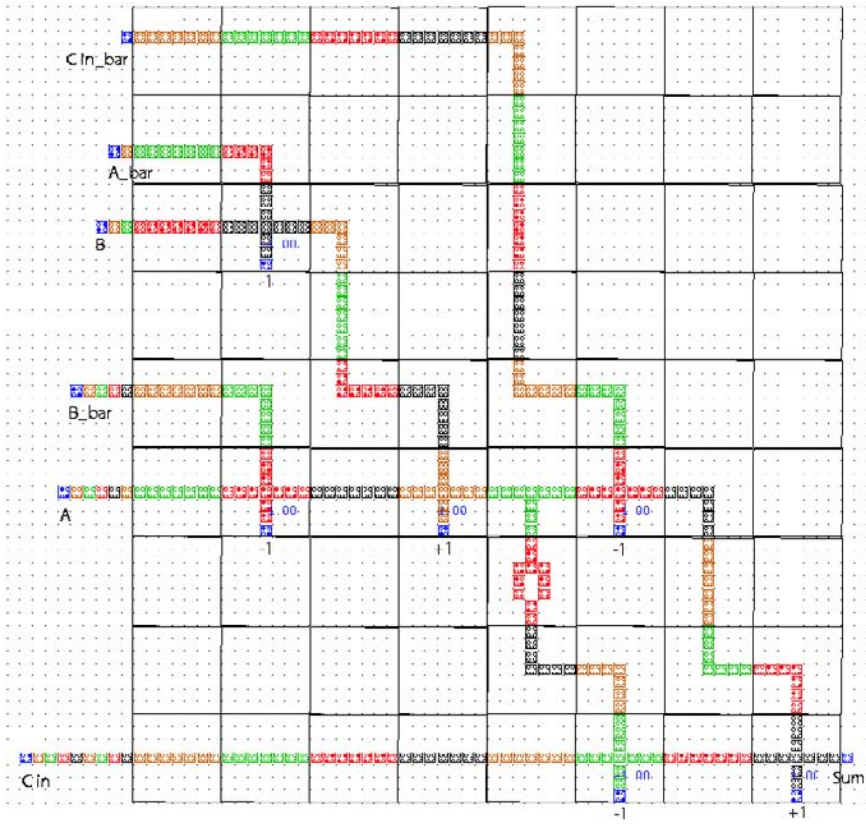


Figure 6.13 One-bit Adder Under 2DD Clocking Scheme for Simulation Using QCADesigner

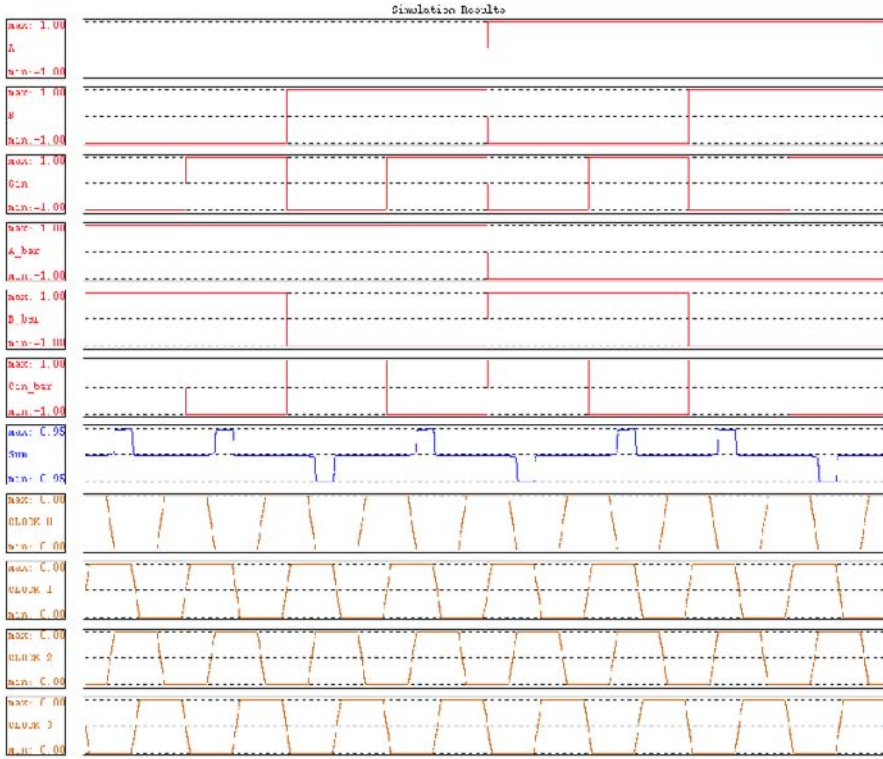


Figure 6.14 Simulation Waveforms for One-bit Adder Under 2DD Clocking Scheme

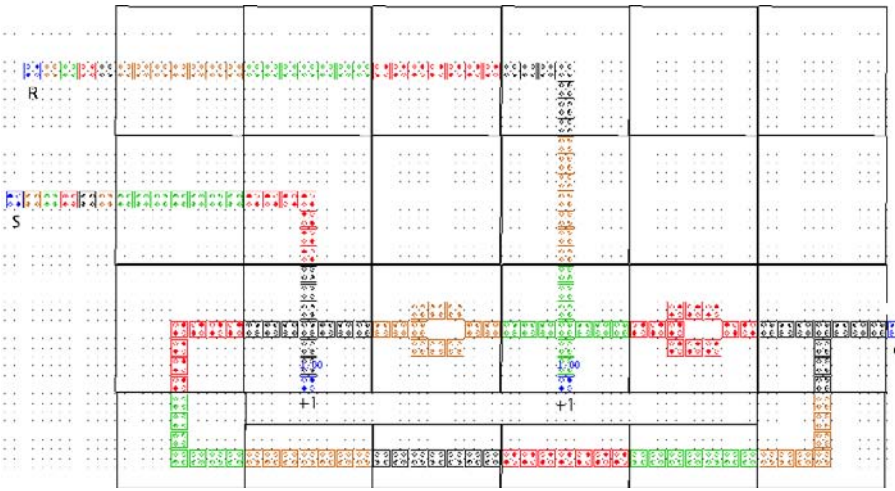


Figure 6.15 RS Flip-flop Under 2DD Clocking Scheme for Simulation Using QCADesigner

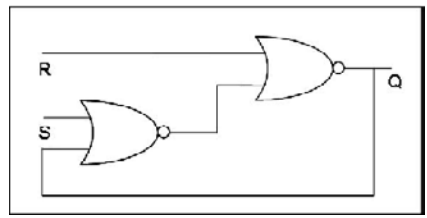


Figure 6.16 Schematic of RS Flip-flop Used in the QCA Design

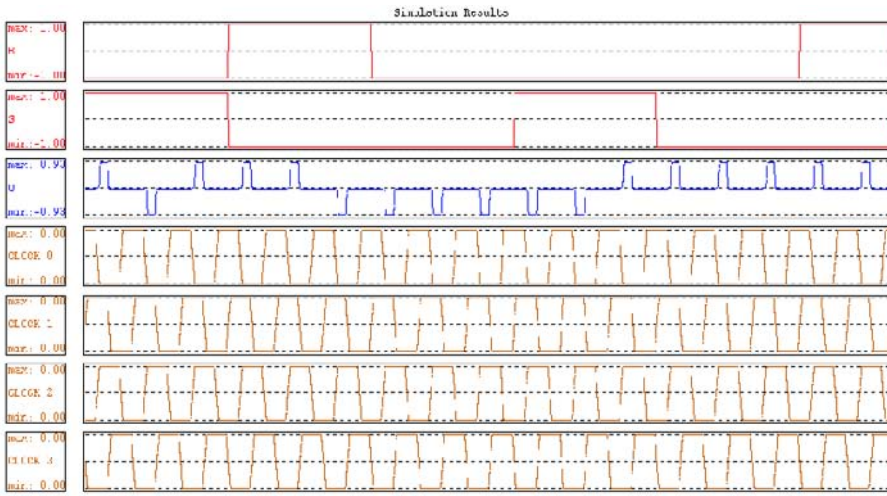


Figure 6.17 Simulation Waveforms for RS Flip-flop Under 2DD Clocking Scheme

performance metrics. The significant reduction in maximum line length permits a fast timing and efficient pipelining to occur, while guaranteeing kink-free behavior in switching. It has been shown that the proposed two-dimensional schemes can also be used in a layout with feedback paths, thus confirming their applicability to sequential circuits implemented by QCA. The proposed clocking schemes have been evaluated by modifying the widely used tool QCADesigner. Combinational and sequential circuits have been presented and evaluated.

References

[1] Lent, C. S. and P. D. Tougaw "A Device Architecture for Computing with Quantum Dots," *Proc. of the IEEE*, Vol. 85, 1997, pp. 541-557.

[2] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also available online: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

[3] Griffiths, D. J., *Introduction to Quantum Mechanics*, Englewood Cliffs, NJ: Prentice Hall, 1994.

[4] Lent, C. S., P. D. Tougaw and W. Porod, "Quantum Cellular Automata: The Physics of Computing with Arrays of Quantum Dot Molecules," *PhysComp '94: Proceedings of the Workshop on Physics and Computing*, IEEE Computer Society Press, 1994, pp. 5-13.

[5] Niemier, M. T. and P. M. Kogge, "Problems in Designing with QCAs: Layout=Timing," *International Journal of Circuit Theory and Applications*,Vol. 29, No. 1, 2001, pp. 49-62.

[6] Kung, H. T. and C. E. Leiserson, ”Systolic arrays (for VLSI),” In *Sparse Matrix Proceedings*, pp. 256-282, I.S. Duff and G.W. Stewart (eds), 1978.

[7] Kung, S. Y., et al., ”Wavefront Array Processor: Language, Architecture, and Applications,” *Special Issue of the IEEE Trans. on Computers and Parallel and Distributed Processing*,Vol.31, No. 11, 1982, pp. 1054-1066.

[8] Lent, C. S. and B. Isaksen, “Clocked Molecular Quantum-Dot Cellular Automata,” *IEEE Transactions on Electron Devices*,Vol. 50, No. 9, 2003, pp. 1890-1895.

[9] Vankamamidi, V., M. Ottavi, and F. Lombardi, “Timing and Clocking of QCA Systems,” Northeastern University, ECE Department, Internal Report 2004 (available upon request).

[10] Vankamamidi, V., M. Ottavi, and F. Lombardi, ”Tile-Based Design of a Serial Memory in QCA,” *Proc. ACM/IEEE Great Lakes Symposium on VLSI*, 2005, pp. 201-206.

[11] Vankamamidi, V., M. Ottavi, and F. Lombardi, ”A Line-Based Parallel Memory for QCA Implementation”, *IEEE Transactions on Nanotechnology*, vol 4, No. 6, 2005, pp. 690-698.

[12] Momenzadeh, M., J. Huang, and F. Lombardi, ”Defect Characterization and Tolerance of QCA Sequential Devices and Circuits,” *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2005, pp. 199-207.

Chapter 7

Tile-Based QCA Design

J. Huang, M. Momenzadeh, L. Schiano, M. Ottavi

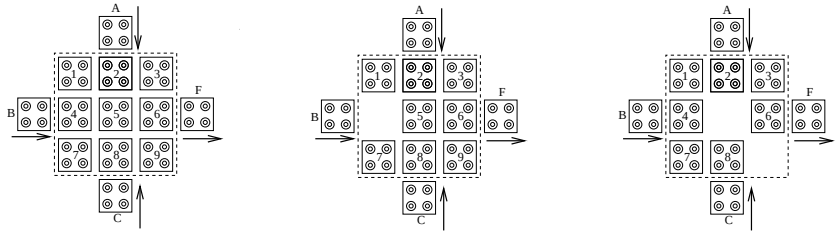
and F. Lombardi

In this chapter, a tile-based modular QCA design methodology is pursued. This approach takes into account the PBW paradigm of QCA, in which information manipulation can be accomplished while transmission and communication of signals take place. PBW capabilities can be observed in the so-called inverter chain as well as in the arrangement of the cells in an MV.

The existing literature on QCA design mostly uses a gate-based methodology [1] [2], as introduced in Chapter 4. In a gate-based design, following logic synthesis, individual gates (MV and INV) are connected to form the desired circuit. The majority function (MV) is not universal, so inversion (INV) is also required. Inversion can be achieved in QCA using a 45 degrees cell orientation. However, it has been shown that this arrangement is not defect-tolerant [3]. An inverter chain (Figure 3.4) can be used. An issue associated with using the inverter chain is that rotated cells (cells rotated by 45 degrees) are employed; these cells are difficult to manufacture. Inversion can also be achieved using the INV gate (Figure 3.3). In CMOS, the INV is the simplest gate. However, in QCA the INV gate is at least as large as the MV.

As explained previously in Chapter 3, recent research in QCA manufacturing focus on molecular implementations. Molecular QCA manufacturing techniques are well suited for modularization through a *structured* QCA design. QCA design can be implemented by modularization through a simple, Manhattan-style interconnect. However, this design is expected to generate an area overhead compared to a gate-based design. This has also been encountered in CMOS: a design using

a full-custom layout is usually smaller than a design using standard-cells. In the technical literature, structured QCA design has not been treated in depth. A modular methodology known as SQUARES has been proposed in [4]. However, this methodology considers cell interactions to occur only within a 5×5 grid and no analysis has been reported on a non fully populated (NFP) grid. Figure 7.1(a) illustrates a 3×3 FP grid (made of 9 cells) with three inputs (A, B and C) and one output (F). An NFP grid is obtained by selectively undepositing some QCA cells from the FP grid. Figures 7.1(b) and (c) illustrate two instances of a 3×3 NFP grid when one and two cells are undeposited, respectively. Each numbered box is a QCA cell that may be deposited (i.e., included in the final QCA layout).



(a) Fully Populated (b) Non Fully Populated (c) Non Fully Populated

Figure 7.1 Examples of Fully and Non Fully Populated QCA Grids

As in the early stages of VLSI, QCA requires building blocks that are versatile to allow flexible manufacturing and assembly of different circuits. In this chapter, a modular approach based on elementary building blocks referred to as tiles, is proposed for QCA design. This method begins by the logic characterization of a set of tiles, which is used as the basic logic building blocks of the circuit. A tile is built using an $n \times n$ square grid of QCA cells. Both fully populated (FP) as well as the non fully populated (NFP) grids are analyzed and used as logic building blocks. For an $n \times n$ grid, there are 2^n different possibilities for depositing cells; each of them is referred to as a configuration. In the next step, grids and input/output cells are integrated into tiles and then into circuits. Finally, the QCA layout as well as clocking zone assignments are generated. This methodology is applicable in the design phase prior to cell deposition. To limit unwanted interactions, isolation among tiles is enforced through the input/output cells and spacing among tiles. It will be shown that the proposed tiles are not only versatile in logic function generation, but also inherently defect tolerant. This methodology results in a tile clocking arrangement that is simpler than the one required by SQUARES.

Of the $n \times n$ grid, the 3×3 grid has shown to have unique properties that make it very attractive for synthesizing and designing larger circuits. Tiles built with the 3×3 grid are therefore used as examples to illustrate the proposed methodology. Using different input and output cells, five tiles are analyzed as they provide a high degree of flexibility in logic operation. Logic characterization as well as defect tolerance of the tiles built with 3×3 grids is investigated in detail in this chapter. Different logic functions can be generated by using less than n^2 cells in a grid of dimension n (NFP grid). New functions such as majority-like functions and wire crossings can be generated by selectively undepositing cells. The majority-like function, which is the MV function with input inversions (e.g., majority voting of A', B and C), is very efficient in logic design since it eliminates the inverter at the input. The functional behaviors of the tiles are reported using simulation and analyzed in detail. Circuits built with the proposed method are compared with SQUARES and gate-based design (in which circuits are built with gates such as MV and INV). It will be shown that this methodology results in a tile clocking arrangement that is simpler than the one required by SQUARES.

The proposed tiles are inherently defect tolerant, and the defect characterization of the five tiles built with 3×3 grids is analyzed in detail. As applicable to molecular QCA, only the undeposited cell defect is considered. This represents the case when the defective cell fails to attach to the substrate. For defect tolerance the relationship between a fault-free tile and a tile with undeposited cells is very important because it defines the paradigm of PBW for a tile-based design. It is evident that a tile inherently offers significant defect tolerance as its nine cells closely interact in a spatial redundancy arrangement.

All simulations are performed using the coherence vector engine of QCADesigner v.1.4.0 simulation tool. Hereafter, the following assumptions are made:

1. Only undeposited cell defects are considered as most likely to occur in molecular implementations.
2. The one-dimensional clocking scheme is assumed [5]; clocking is from left (inputs) to right (outputs). So, all cells in a tile (grid and input/output cells) are assumed to be within a single timing zone and all tiles in the same column are in the same zone.
3. The no logic state (called the *undetermined state* and denoted by "-") may occur for some patterns due to lack of definitive polarization at the output.
4. In all simulations, the following parameters are used: cell size is $10 \times 10nm^2$, the cell-to-cell distance is $2.5nm$. The dot has $2.5nm$ diameter. The radius of

effect R is set to $40nm$. Unless otherwise noted, a combinatorially exhaustive evaluation of the tiles and grids is pursued.

7.1 QCA DESIGN BY TILING

The design of nano circuits and systems requires a substantially different approach than those used in CMOS-based VLSI. The large density expected for QCA (especially for molecular implementations) [6] [7] represents one of the significant features in the manufacturing of these systems. However, as a technology in infancy, QCA still requires versatile building blocks for deposition and circuit assembly. The SQUARES methodology has been proposed [4], in which the basic building block is a 5×5 QCA cell grid. Logic functions are determined based on the direct embedding of the MV and INV circuits into this grid, rather than on analyzing the different interactions and possible configurations of the QCA cells. Therefore, the methodology of SQUARES is very restrictive and can result in wasted area once cells are deposited on a substrate. Further, timing is also complex in the SQUARES approach. In this chapter, a new methodology is proposed for QCA. This methodology partially relies on the early work of [4], but it provides a more complete characterization of the design process at logic level. The proposed approach not only analyzes the role of a grid with respect to the generation of logic functions (inclusive of the coplanar wire crossing), but also it assesses the effects of input and output cells (and corresponding signals) on the operation of the circuit as a whole prior to cell deposition on a substrate.

Since it can be implemented within a CAD framework, tiling is very relevant for an emerging technology such as QCA. However for CAD implementation, the following features must be properly addressed:

- Tiles must be flexible in the generation of logic functions at high polarization levels. As explained in Chapter 3, the two fully polarized states represent the two logic states in QCA. Moreover, tiles should also be robust to limit interactions from unwanted cells.
- Between and among tiles, signals should be routed at ease (such as with a Manhattan strategy).
- The tiles should have stable signals; that is, no undetermined value (due to lack of polarization or the presence of a glitch) should be present at an output.

The proposed design methodology is applicable prior to cell deposition and can be described as follows. A flow chart of the proposed method is shown in Figure 7.2. First a set of tiles are chosen as the basic building blocks of the circuit. Let a *tile* be defined as a square cell grid (FP or NFP) with the addition of input and output cells. An input or output cell can be placed only in the middle of each side of the grid. A detailed logic characterization is then performed to establish the logic capabilities of the tiles (i.e., the unique logic functions that can be realized by the tiles). This can be accomplished by simulation. For each logic function, there may be more than one possible configuration. The best configuration is chosen based on design requirements. For each logic function, the tile with the best configuration is chosen. After the logic behavior of all the tiles are known, these tiles are assembled into desired circuit in the logic mapping step. Clocking zone assignments are determined. Finally, the QCA layout is obtained and manufacture of the circuit can be done by depositing the QCA cells present in the layout to the substrate.

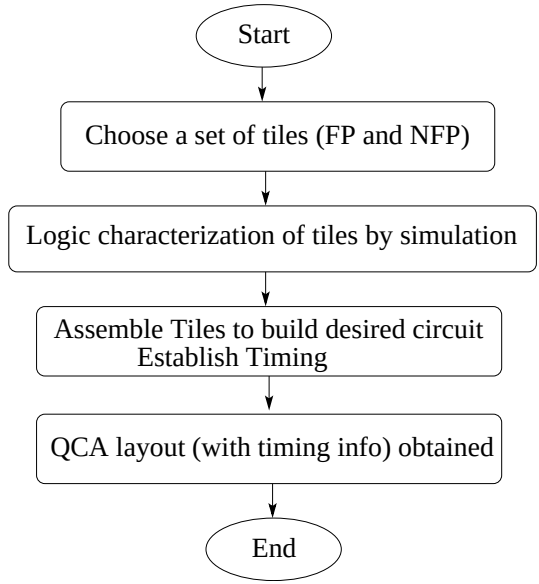

Start
Choose a set of tiles (FP and NFP)
Logic characterization of tiles by simulation
Assemble Tiles to build desired circuit
Establish Timing
QCA layout (with timing info) obtained
End

Figure 7.2 Flowchart of Proposed Methodology

The logic mapping step is discussed in detail next. Consider a two-dimensional square matrix A of dimension N; A can be thought to represent the QCA layout (at logic level) prior to cell deposition on a substrate. Within this layout, two sets of

tiles can be used: active and passive tiles. A tile is defined as active if it implements a combinational logic function with minterm(s) (of at least two literals). A tile is said to be passive if it implements logic functions of only one literal, i.e., wire or INV function. The proposed technique relies on some tiles (mostly active tiles) to perform logic computation, while passive tiles perform limited computation. Passive tiles are mostly used for signal transfer/routing as well as providing separation among active tiles in the layout. Unwanted interactions among cells in different grids are very limited. Moreover, no immediate adjacency between tiles is allowed. Hence, isolation is enforced through spacing (as provided by an area with no cells between tiles). Following logic synthesis and technology mapping (by which combinational functions of a QCA circuit S are mapped to specific QCA gates, such as MVs), then the problem of logically customizing the layout A for implementing S can be thought of as the iterative execution of the following basic steps:

1. Divide A into K^2 square grids (of dimension n) where the integer K is given by $K = \frac{N}{n}$.
2. Sequentially map each logic function into a tile. If there is an adjacency (either vertically or horizontally) among the grids, then in addition to the input and output cells, spacing (an area with no cells) is inserted between grids, thus preventing unwanted interactions.
3. Route the signals using passive tiles. If not successful go back to Step 2 above using a different mapping.

In the above process, clocking issues must also be included, i.e., S is partitioned into zones to preserve the correct flow of data [8]. Without loss of correctness or generality, S is assumed to be expressed in SOP (Sum Of Products) form with minterms of possibly multiple literals; many iterations may be required for successful routing using passive tiles.

Note that as A is generated prior to cell deposition, then only those cells which are required for implementing S, are kept in the final layout.

7.2 FULLY POPULATED GRID ANALYSIS

The first issue associated with a tile-based design for QCA is to establish its logic capabilities as related to the dimension of a grid (i.e., n). For QCA logic design by existing techniques, two basic devices can be used:

- An MV with 3 inputs (A,B,C) and one output F, where F = Maj(A,B,C) = $AB + AC + BC$.
- An INV with one input A and output F ($F = A$').

The correct operation of these devices in larger circuits is based on the assumption that interactions with cells in other INVs and MVs are negligible (if multiple devices are present in the same clocking zone). This is not always applicable to a QCA circuit and no work has been reported on establishing the close interactions of QCA cells within an FP grid. Moreover, generation of logic functions (differently from MV and INV) is possible by exploiting the spatial arrangement of the cells in the Cartesian plane (such as in a NFP grid).

In the absence of an analytical framework, in this chapter an exhaustive simulation of square FP grids of different dimension (i.e., by varying n) is employed for establishing the logic functions at the grid output(s). Initially due to the exponential complexity in the number of combinations for the inputs and outputs to an $n \times n$ grid, a simpler evaluation has been pursued based on the following arrangement: for an $n \times n$ fully populated grid, only n inputs and n outputs are assumed. This arrangement is consistent with the one-dimensional clocking technique commonly used for QCA; in this case, 2^n input patterns must be simulated.

Different square FP grids with dimension from 2 to 6 have been evaluated by simulation. No input or output cell is attached such that the logic output operations are assessed based on the cells in the grid, that is, no interaction from cells external to the grid can occur (this assumption will be removed once input/output cells are added as part of a tile). Inputs are labelled in alphabetical order (from top to bottom on the left), while outputs are denoted by F_i (in increasing index i from top to bottom on the right). For example, the 3×3 FP grid is shown in Figure 7.3(a), while the 4×4 FP grid is shown in Figure 7.3(b).

For each of the considered FP grids, the complete truth tables have been obtained by exhaustive simulation with QCA Designer and the minimized output functions have been obtained by using Espresso as synthesis tool.

- For the 2×2 grid, the output functions are $F1 = A$, $F2 = B$. Thus, this grid behaves like a wire for both outputs (i.e., it is passive).
- The 3×3 grid has the following output functions: $F1 = A$, $F2 = AB + AC + BC$, and $F3 = C$. Hence, the upper and lower outputs $F1$ and $F3$ behave as wires, while the center output $F2$ acts as an MV.
- For the 4×4 grid, some outputs have an undetermined value as the simulator showed unstable (oscillating) values for some inputs' combinations. The

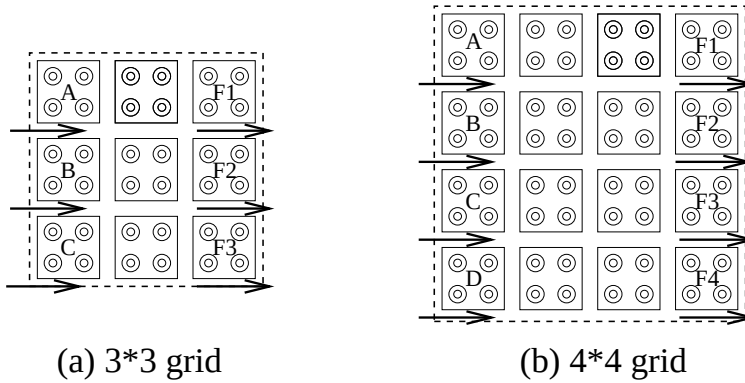


Figure 7.3 (a) 3×3 FP Grid and (b) 4×4 FP Grid

output functions (assuming a 1 for the undetermined values) are as follows: $F1 = A$, $F2 = AB + BC + AC$, $F3 = CD + BD + BC + AC$, and $F4 = D$. Also in this case, the upper and lower outputs $F1$ and $F4$ behave as wires, while the output $F3$ is an SOP. However, the presence of undetermined values in the outputs severely limits the use of this grid.

- In contrast with the 4×4 grid the 5×5 grid has no undetermined value in the outputs. The output functions are as follows: $F1 = A$, $F2 = ACD + BDE + AB + ACE + BC$, $F3 = BDE + ABD + DE + ACE + CD$, $F4 = BCE + ABD + BC + ACE + CD$ and $F5 = E$. Also in this case, the upper and lower outputs $F1$ and $F5$ behave as wires, while the functions $F2$, $F3$ and $F4$ are in SOP form.
- The 6×6 grid has no undetermined value, the output functions are as follows: $F1 = A + BCD'E'F'$, $F2 = ACD + A'BDF' + ACF' + AB + BC$, $F3 = ABDF + ABDE + A'BCF' + ACF' + BC + CD$, $F4 = B'CEF' + BCEF + A'BDF' + A'CEF' + DF + DE + CD$, $F5 = B'CEF' + A'CEF' + EF + DF + DE$ and $F6 = F$. In this case, the obtained functions have two unique features compared to other grids: $F1$ does not behave as a wire; literals appear also in an inverted form.

It is evident that the 2×2 and the 4×4 grids cannot be utilized due to the lack of polarization (in the latter) and processing (in the former). As for the 5×5 and 6×6 grids, the output functions are rather complex and do not map efficiently to a logic

synthesis process due to the irregular nature of the SOP and the minterms (with different numbers of literals). Moreover, it has been shown that for logic design, functions of more than 4 literals in a minterm seldom occur in practice [9]. So in this chapter, the 3×3 grid has been selected for an in-depth tile analysis; this is consistent with the condition that the side of the grid should have an odd number of cells to allow the placement of an input or output cell at its center.

This result has also an impact on the area that a circuit occupies in the layout: it is obvious that a 3×3 FP grid can embed an MV or an INV (i.e., with no input/output cell, the MV and INV are isomorphic to the 3×3 grid). So while the number of cells may be larger in a tile-based design, the layout area is not increased compared to a gate-based technique for QCA design that utilizes discrete devices, such as MV and INV.

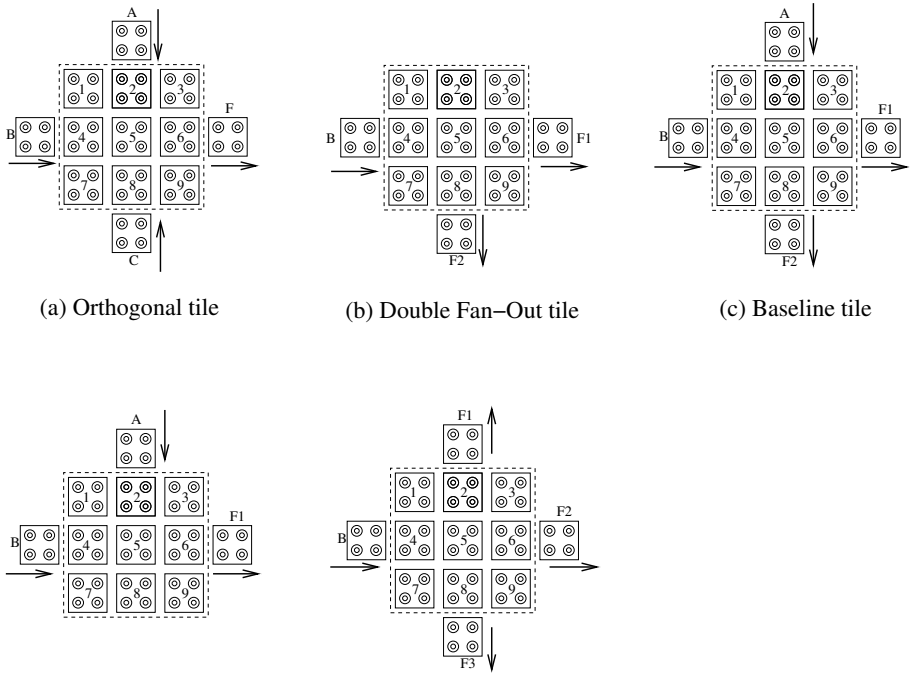
7.3 TILES BASED ON 3×3 GRIDS

A non fully populated grid is generated from a fully populated grid by selectively undepositing cells. This process changes the logic behavior of a QCA circuit; moreover, only the cells that are kept in the final layout are deposited on the substrate. So, it is interesting to compare the characteristics of tiles with different input/output cells. For the 3×3 grid, the following tiles (shown in Figure 7.4) are possible. Tiles with one input and one output are not considered due to the obvious wire function; they are referred to as interconnection (passive) tiles. Let U denote the set of undeposited cells (as labelled in Figure 7.4) and F denote the generated output function. For example, in Table 7.1 Maj(A', B, C) is achieved as output function if cell 2 is not deposited in an orthogonal tile.

7.3.1 Orthogonal Tile

The orthogonal tile is shown in Figure 7.4(a). This tile has three inputs (the horizontal input cell B and the vertical input cells A and C) and one output (the horizontal output cell F). In the defect-free case, the output of this tile is Maj(A,B,C)=$AB + BC + AC$. Thus, this is the basic logic block in the tile-based design of QCA and its defect-tolerant properties are very important to assess. Table 7.1 shows the simulation results when at most one cell is undeposited from the orthogonal tile. The probability of generating different majority functions versus number of undeposited cells is shown in Figure 7.5.

These new MV-like functions (with input inversion in the MV) are possible due to the interaction of the cells at the corners of the tile with the center cell of



(a) Orthogonal tile (b) Double Fan–Out tile (c) Baseline tile

(d) Fan–In tile (e) Triple Fan–Out tile

Figure 7.4 3×3 Tiles with an FP Grid

Table 7.1
Generation of Output Function by Undepositing at Most One Cell in the Orthogonal Tile

U	F	U	F
none	Maj(A, B, C)	5	Maj(A, B, C)
1	Maj(A, B, C)	6	Maj(A', B, C')
2	Maj(A', B, C)	7	Maj(A, B, C)
3	Maj(A, B, C)	8	Maj(A, B, C')
4	Maj(A, B, C)	9	Maj(A, B, C)

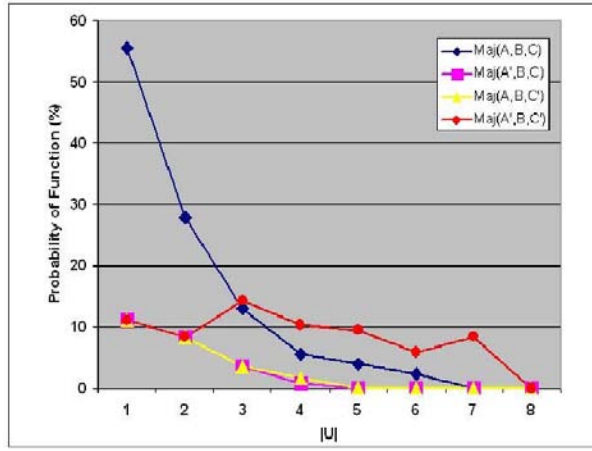


Figure 7.5 Probability of Generating Different Majority Functions versus Number of Undeposited Cells in the Orthogonal Tile

the MV (i.e., cell 6). As an example, Figure 7.6 presents a comparison between the tile-based design of this so-called MV-like function Maj(A', B, C) with designs obtained by SQUARES and using QCA devices in a traditional gate-based method. The tile-based design of this function requires an area (with no input/output cell) of 9 cells, while the SQUARES-based and gate-based designs require areas of $6 \times 25 = 150$ and $7 \times 8 = 56$ cells, respectively. Additionally, the tile-based design has a smaller delay compared to the SQUARES-based and gate-based designs. One clock zone is needed in the tile-based design, while 2 and 3 clock zones are used in the SQUARES-based and gate-based designs. MV-like functions provide a significant degree of freedom in designing QCA circuits. For example, an MV-like function with two complemented variables can implement a two-input NAND or NOR gate (thus saving on the number of QCA cells as compared to a traditional design that utilizes an MV and an INV).

An exhaustive simulation has also been pursued for tiles with NFP grids, that is, the absence of i undeposited cells, $i = 1, 2....8$ from the layout (note that for all tiles the absence of all cells results in all outputs to take an undetermined value prior to deposition on a substrate). For the orthogonal tile, the number of patterns of each output function when i cells are undeposited is shown in Table 7.2.

The following observations can be made from the simulations:

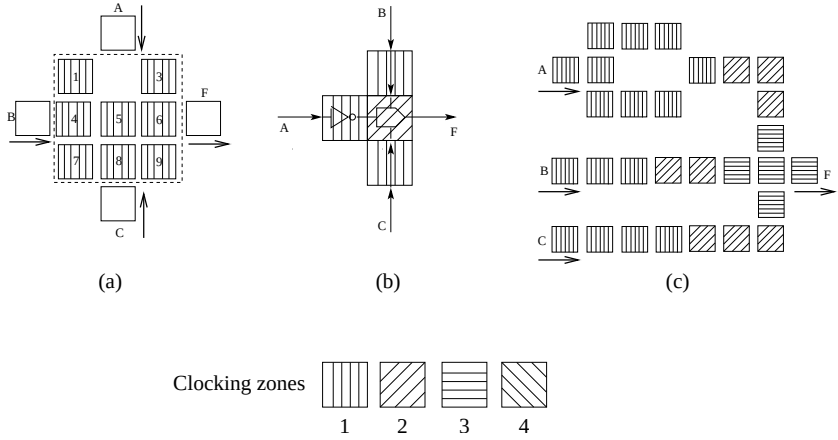


Figure 7.6 Design of the Maj(A', B, C) Function by (a) Tile-based, (b) SQUARES-based, and (c) Gate-based Methods

Table 7.2

Number of Occurrences of Output Functions in the Orthogonal Tile by Undepositing $|U|$ Cells

F	$\|U\|$							
	1	2	3	4	5	6	7	8
A	0	0	2	5	9	9	5	1
A'	0	3	13	28	28	15	3	0
B	0	6	12	13	11	6	1	0
C	0	0	2	5	7	10	5	1
C'	0	3	15	28	30	15	3	0
-	0	4	11	24	24	22	16	6
Maj(A, B, C)	6	10	11	7	5	2	0	0
Maj(A', B, C)	1	3	3	1	0	0	0	0
Maj(A, B, C')	1	3	3	2	0	0	0	0
Maj(A', B, C')	1	3	12	13	12	5	3	0
NXOR	0	1	0	0	0	0	0	0
Total	9	36	84	126	126	84	36	9

1. The FP orthogonal tile behaves as an MV. In almost all cases, a NFP orthogonal tile (FP orthogonal tile with undeposited cells) behaves in the following two ways: wire/inverting functions or MV/MV-like functions.
2. The MV-like functions provided by NFP orthogonal tile are versatile in logic design.
3. Assume the FP orthogonal tile is the fault free tile and undeposited cell defects may occur. Then the tile is fault tolerant and in most cases can still perform stable and useful logic functions. Additionally, undeposited cell defects that occur in the corner cells (cells 1, 3, 7 and 9) do not change the logic function of the tile, thus confirming the non-defect tolerant design of an MV.
4. In the simulations using the coherence vector engine, whenever cell 6 is undeposited, the polarization level experiences a drop. In all simulated occurrences with cell 6 present, the magnitude of the maximum polarization is above ± 0.9. However, when cell 6 is undeposited, then the magnitude of the maximum polarization level drops below ± 0.77. When other additional cells are undeposited (besides cell 6), in many cases the polarization level for some input patterns is so low that no definite logic function can be observed at the output.

The average magnitude of the maximum polarization level of the output when a number of cells is undeposited is shown in Figure 7.7. As for previous tiles, in some cases the output exhibits no definite polarization level. However, in some of these cases the polarization level is quite high. This is due to the fact that for some cases, only some of the input patterns cause no or very low polarization, while other input patterns give definite outputs with high polarization levels. Also, when increasing the number of undeposited cells as defects, the decrease in polarization level is not significant.

7.3.2 Double Fan-out Tile

The double fan-out tile, as shown in Figure 7.4(b) is analyzed next. The double fan-out tile has one input (provided by the horizontal cell B) and two outputs (i.e. the horizontal output cell $F1$ and the vertical output cell $F2$). In the FP double fan-out tile, both outputs follow the value of the input cell, that is, $F1 = F2 = B$ (wire function), hence the tile behaves as a fan-out point in a CMOS interconnect. Table 7.3 shows the simulation results when at most one cell is undeposited from

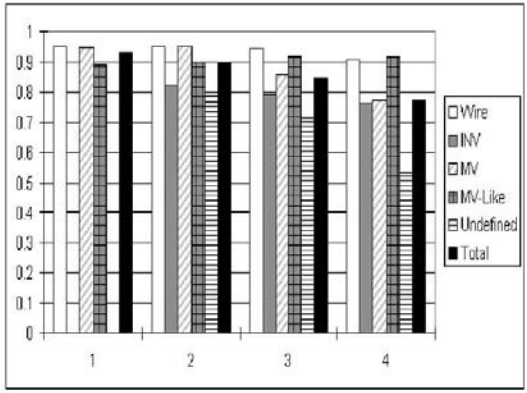


Figure 7.7 Average Magnitude of the Maximum Polarization Level of Orthogonal Tile

Table 7.3
Generation of Output Function by Undepositing at most a Cell in the Double Fan-out Tile

U	$F1$	$F2$	U	$F1$	$F2$
none	B	B	5	B'	B'
1	B	B	6	B'	B
2	B	B	7	B	B
3	B	B	8	B	B'
4	B	B	9	B	B

the double fan-out tile. The exhaustive simulation depicts nine output functions as shown in Table 7.4 for the double fan-out tile.

It can be observed that for the NFP tile in the presence of multiple undeposited cells, in most cases the tile produces either a wire function, or an inverting function (the output is the complement of the input). The probability of generating different functions for the two outputs ($F1$ and $F2$) is summarized in Figure 7.8. Even with four undeposited cells due to defects, in almost 90% of the cases the tile can still function either as a wire, or an inverter due to its spatial redundancy, thus providing an excellent level of functionality. The following additional observations can be drawn:

Table 7.4
Number of Occurrences of Output Functions in the Double Fan-out Tile by Undepositing $|U|$ Cells

$F1F2$	$\|U\|$							
	1	2	3	4	5	6	7	8
BB	6	12	17	25	30	19	3	0
$B'B'$	1	10	17	17	9	3	1	0
- -	0	0	0	1	2	5	9	6
BB'	1	7	27	54	48	19	3	0
$B'B$	1	6	13	10	7	2	0	0
B-	0	0	5	10	10	15	8	1
-B	0	0	1	5	6	8	5	1
B'-	0	0	2	2	4	1	0	0
-B'	0	1	2	2	10	12	7	1
Total	9	36	84	126	126	84	36	9

1. The probability of being in an undefined state for an output signal increases with the number of undeposited cells. Moreover, such probability is greater at $F2$ than at $F1$ once the number of defects is more than 2.

2. The probability of having a wire function in the horizontal output $F1$ is greater than for the vertical output $F2$, indicating that signal propagation in the one-dimensional clocking scheme is stronger along the direction of signal flow (perpendicular to the direction of the underlying E field).

3. The probability of having an inverting function in the vertical output $F2$ is greater than for the horizontal output $F1$. This is expected due to the 90^o orientation of the output cell with respect to the input cell and the possible 45^o misalignments in the defect-free cells.

4. The polarization plots for two extreme cases are shown in Figures 7.9(a) (no defect) and 7.9(b) (all nine cells undeposited). In the defect-free case, both outputs exhibit the wire function with high polarization levels. In the extreme case when all cells in the grid are undeposited, logically $F1$ still produces the wire function, while $F2$ performs the inverting function; however, both outputs have a very low polarization level (below the 0.1 value), thus an undefined state function is generated.

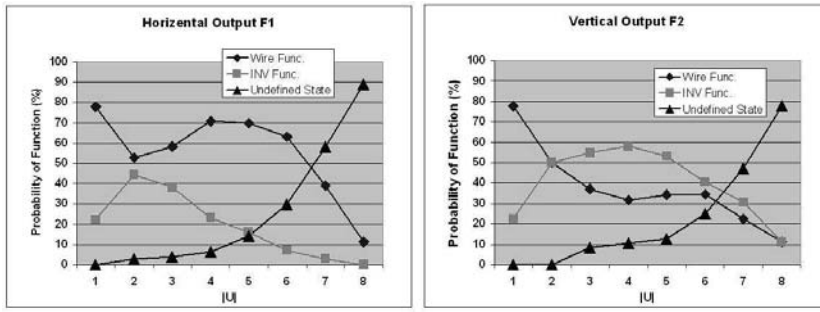


Figure 7.8 Probability of Generating Different Functions vs. # of Undeposited Cells in the Double Fan-out Tile

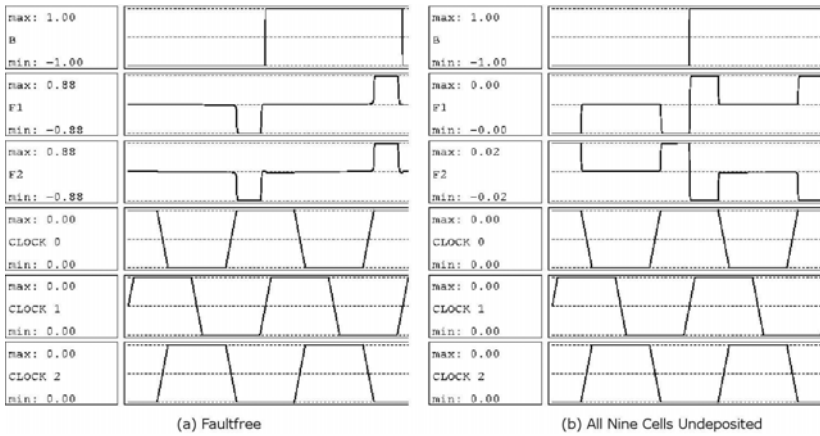


Figure 7.9 Polarization Plot of Fan-out Tile

The average values of the maximum polarization level (magnitude) at the outputs ($F1$ and $F2$) are shown in Figure 7.10. The diagram shows the average magnitude of the maximum output polarization level when undeposited cells are incrementally present. The average magnitude of the maximum polarization level is reported for the wire function, the inverting function and the undefined state function, as well as the total. The polarization level of the wire function is higher than the inverting function in all cases, thus confirming that this tile provides excellent defect-tolerant capabilities compared to other functional behaviors due to defects.

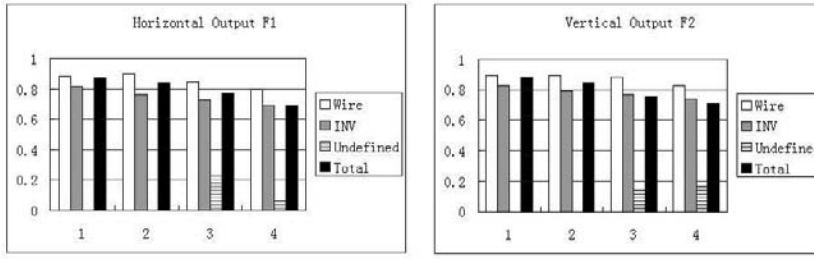


Figure 7.10 Average Magnitude of the Maximum Polarization Level of Fan-out Tile

The double fan-out tile acts as a fan-out point in the circuit, with each of its outputs either following the input or the inversion of the input.

7.3.3 Baseline Tile

The baseline tile is shown in Figure 7.4(c); it has two inputs (one vertical input cell A and one horizontal input cell B) and two outputs (the horizontal output cell $F1$ and the vertical output cell $F2$). The FP baseline tile accomplishes coplanar wire crossing, i.e., $F1F2 = BA$. Table 7.5 shows the simulation results when at most one cell is undeposited from the baseline tile.

For the baseline tile, altogether 20 output functions are observed during exhaustive simulation, such as $F1F2 = AB$, AB', ... The number of occurrences for these functions when undepositing a specific number of cells, is reported in Table 7.8. Note that by undepositing cells no additional inversion on the input signals is generated (while still retaining the crossing property), i.e. $F1F2 = BA', B'A, B'A'$ are not observed in the simulation. In the fully populated case this tile works as a switch (or coplanar crossing) with $F1F2 = BA$, in which the two input signals cross each other. For the NFP baseline tile, three types of

Table 7.5
Generation of Output Function by Undepositing at Most One Cell in the Baseline Tile

U	$F1$	$F2$	U	$F1$	$F2$
none	B	A	5	A'	B'
1	B	A	6	A'	A
2	B	B	7	A	A
3	B	B	8	B	B'
4	A	A	9	B	A

functions are observed. The first type is the coplanar crossing, which is also the fault free logic function. The second type is when the tile works as two L-shape wires, with $F1F2 = AB$. Also included in the second type are the L-shape wires with inversions, such as $F1F2 = A'B$, $F1F2 = A'B'$. The third type is the fan-out, when the two outputs follows the same input, such as $F1F2 = BB$, $F1F2 = AA$. Also included in the third type is fan-out with inversions, such as $F1F2 = B'B$. The last type is the undefined, where at least one output exhibits the undetermined state. The probability of having these function types versus the number of undeposited cells is plotted in Figure 7.11.

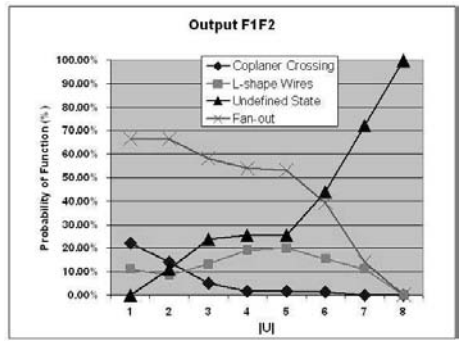


Figure 7.11 Probability of Generating Different Functions vs. # of Undeposited Cells in the Baseline Tile

The average of the maximum polarization level (magnitude) of the outputs is shown in Figure 7.12. The results also show symmetry in the operation of this tile; for example, when a number of cells are removed, the probability of

Table 7.6
Number of Occurrences of Output Functions in the Baseline Tile when Undepositing $|U|$ Cells

$F1F2$	$\|U\|$							
	1	2	3	4	5	6	7	8
BA	2	5	4	2	2	1	0	0
$A'B'$	1	3	8	18	10	2	1	0
BB	1	5	9	7	7	5	2	0
BB'	1	5	12	22	21	11	1	0
$A'A$	1	5	14	22	21	10	0	0
AA	3	5	6	7	7	5	2	0
$A'A'$	0	2	3	4	5	1	0	0
A'-	0	1	3	5	5	7	4	1
A-	0	1	5	5	3	4	3	1
$B'B'$	0	2	3	4	5	1	0	0
-B'	0	1	4	5	5	7	4	1
-B	0	1	3	5	3	4	2	1
AA'	0	0	2	2	1	0	0	0
AB'	0	0	2	2	5	4	1	0
B-	0	0	2	5	6	4	3	0
$A'B$	0	0	1	2	5	4	1	0
$B'B$	0	0	1	2	1	0	0	0
-A	0	0	2	4	6	5	2	0
AB	0	0	0	2	5	3	1	0
- -	0	0	0	1	3	6	8	5
Total	9	36	84	126	126	84	36	9

Table 7.7
Generation of Output Function by Undepositing at Most One Cell in the Fan-in Tile

U	F	U	F
none	B	5	B
1	B	6	A'
2	B	7	B
3	B	8	B
4	A	9	B

generating $F1 = A$ is nearly the same as $F2 = B$. Moreover the average maximum polarization level of A (B) is stronger at $F2(F1)$.

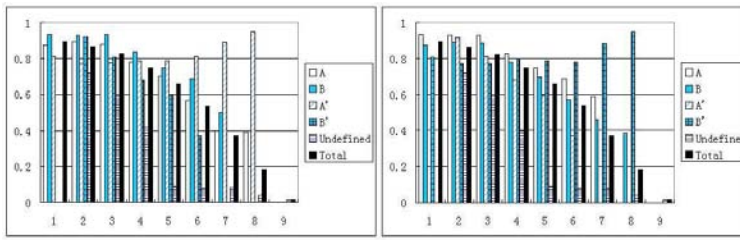


Figure 7.12 Average Maximum Polarization Magnitude of Baseline Tile

7.3.4 Fan-in Tile

The fan-in tile is shown in Figure 7.4(d); this tile has two inputs (one vertical input cell A and one horizontal input cell B) and one output (given by the horizontal output cell F). The simulation results when at most one cell is undeposited, are shown in Table 7.7.

The exhaustive simulation depicts that altogether five output functions are possible in the fan-in tile, namely A, B, A', B' and $-$. The number of occurrences for each of these functions when undepositing a specific number of cells, is reported in Table 7.8. It can be observed that the output either follows A (with possible inversion) or follows B (with possible inversion) or exhibits no definite polarization. These results are summarized in Figure 7.13.

Table 7.8
Number of Occurrences of Output Functions in the Fan-in Tile When Undepositing $|U|$ Cells

F	$\|U\|$							
	1	2	3	4	5	6	7	8
A	1	6	13	17	23	19	6	1
B	7	16	25	36	35	20	5	0
A'	1	10	34	52	46	23	7	1
B'	0	2	5	8	5	0	0	0
-	0	2	7	13	17	22	18	7
Total	9	36	84	126	126	84	36	9

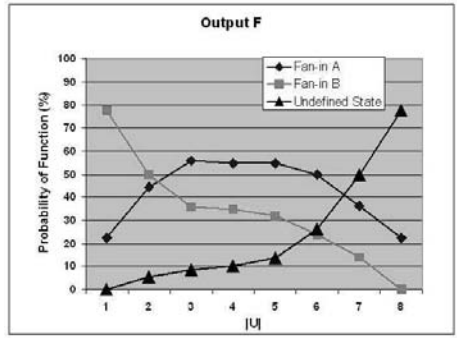


Figure 7.13 Probability of Generating Different Functions vs. # of Undeposited Cells in the Fan-in Tile

Table 7.9
Generation of Output Functions by Undepositing at Most One Cell in the Triple Fan-out Tile

U	$F1$	$F2$	$F3$	U	$F1$	$F2$	$F3$
none	B	B	B	5	B'	B'	B'
1	B	B	B	6	B	B'	B
2	B'	B	B	7	B	B	B
3	B	B	B	8	B	B	B'
4	B'	B'	B'	9	B	B	B

The average maximum polarization level (magnitude) of the output F is shown in Figure 7.14.

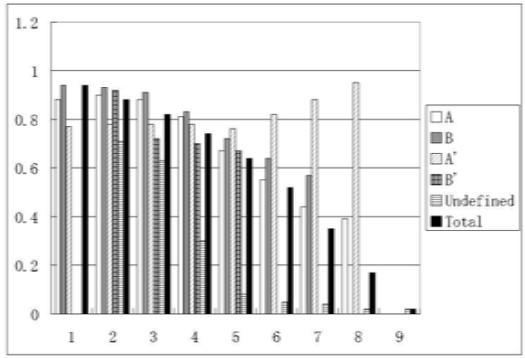


Figure 7.14 Average Maximum Polarization Magnitude of the Fan-in Tile

7.3.5 Triple Fan-out Tile

The triple fan-out tile is shown in Figure 7.4(e); this tile has one input (the horizontal input cell B) and three outputs (the three cells $F1$, $F2$ and $F3$). The simulation results when at most a single cell is undeposited from a triple fan-out tile, are given in Table 7.9.

Various output functions can be observed for the triple fan-out tile, such as $F1F2F3 = BBB, B'BB, ...$ The number of each of these functions when undepositing a specific number of cells, is reported in Table 7.10.

As in the case of the double fan-out tile, in most cases even when multiple cells are undeposited, the tile produces either a wire function or an inverting function. The probability of generating these functions versus the number of undeposited cells is plotted in Figure 7.15. It can be concluded that the probability of output being in undefined state increases with increased number of undeposited cells. Additionally, the probability of having a wire functions at the horizontal input $F2$ is greater than for a vertical input ($F1$ or $F2$) while the probability of having an inverting function for a vertical input ($F1$ or $F2$) is greater than for the horizontal input $F2$.

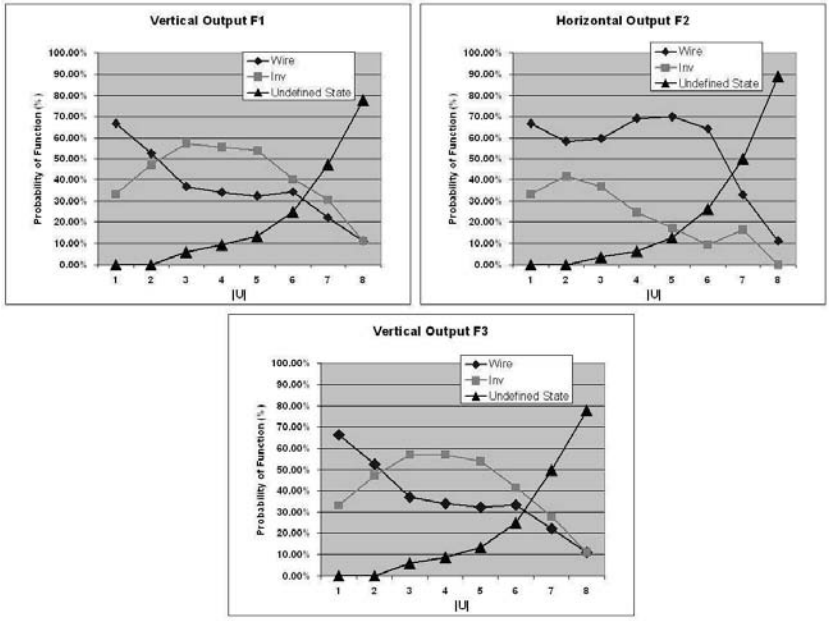


Figure 7.15 Probability of Generating Different Functions vs. # of Undeposited Cells in the Triple Fan-out Tile

Table 7.10
Number of Occurrences of Output Functions in the Triple Fan-out Tile When Undepositing $|U|$ Cells

$F1F2F3$	$\|U\|$							
	1	2	3	4	5	6	7	8
BBB'	1	5	10	11	10	5	0	0
$B'B'B'$	2	7	11	8	7	0	1	0
$BB'B'$	0	2	6	6	1	0	1	0
BBB	4	8	7	8	14	11	0	0
$B'BB$	1	5	10	10	9	5	0	0
$B'B'B$	0	2	5	6	2	0	1	0
$BB'B$	1	4	6	5	6	0	0	0
$B'BB'$	0	3	18	40	35	6	0	0
B-B'	0	0	0	0	3	1	0	0
B-B	0	0	1	6	1	4	1	0
-BB'	0	0	0	3	6	9	3	0
$B'B'$-	0	0	2	2	2	2	3	0
BB'-	0	0	0	1	1	2	0	0
-$B'B'$	0	0	1	2	2	2	0	0
-$B'B$	0	0	0	1	1	2	0	0
-BB	0	0	2	6	3	4	3	0
$B'B$-	0	0	0	3	6	9	0	0
BB-	0	0	1	5	3	4	3	0
B'-B'	0	0	2	1	3	11	1	0
B'-B	0	0	0	1	3	0	1	0
- -B	0	0	0	0	2	2	2	1
B'- -	0	0	0	0	1	1	4	1
- -B'	0	0	0	0	1	1	4	1
B- -	0	0	0	0	2	2	3	1
-B-	0	0	2	0	2	1	3	1
- - -	0	0	0	0	0	0	2	4
Total	9	36	84	126	126	84	36	9

7.4 ANALYSIS OF RESULTS

The following have been observed from the simulation results:

1. In all tiles, the total average magnitude of the maximum polarization level decreases by increasing the number of undeposited cells. While there is little difference between the cascade and fan-out tiles, the orthogonal tile presents a higher level of total polarization. This is caused by the placement of the three inputs and the dominant majority nature of PBW in this tile.
2. In all considered tiles, the probability of no polarization increases when increasing the number of undeposited cells. This is expected because no polarization will be encountered due to the large inter-cell spacing.
3. The tiles provide versatile logic functions that can be used in constructing QCA combinational logic circuits. The MV-like function in orthogonal tile combines the logic function of MV and INV and can be used efficiently in tile-based logic design. As shown in the examples in this chapter, the tile-based design results in reduced area and delay compared with the SQUARES methodology.
4. The double fan-out tile and triple fan-out tile have similar defect tolerance properties (see Figure 7.8 and Figure 7.15). The fault-free function in these tiles is the wire function. It can be observed that with one undeposited cell, the probability of having the correct wire function at the outputs larger than 75% for the double fan-out tile and larger than 65% for the triple fan-out tile. In both these tiles, the probability of obtaining the correct wire function at the horizontal output is greater than the probability of obtaining the correct wire function at the vertical output(s). Even with multiple undeposited cells, in most cases the tiles still produce stable logic function: either the wire function or the inverter function. These functions are very useful in the logic design.
5. The baseline tile acts as a switch (coplanar crossing) in fault-free conditions. However, this switching function is not very fault tolerant. Even with only one undeposited cell, the probability of having switching function is less than 25%. With multiple undeposited cells, in most cases this tile acts as a fan-out (with possible inversion) tile, where the two outputs follow the same input.
6. The presence of new logic functions (such as the inverting function in the fan-out tile, the MV-like functions in orthogonal tiles, or the fan-out function

in baseline tile) shows that defect tolerance can be utilized in accomplishing PBW even under a large number of defective cells and low yield levels.

7.4.1 Configuration Selection

As mentioned earlier, in most cases, a specific logic function can be generated with a number of different configurations of the same tile type. For instance, as shown in Table 7.1, the configurations in which only cell C_1 is undeposited and the configurations in which only cell C_7 is undeposited produce the same logic function of $F = Maj(A, B, C)$. For a specific tile type, the *equivalent-set* is defined as a set of tile configurations that realizes a particular logic function. To select the best configuration for a specific logic function, the configurations of that equivalent set should be ranked according to predefined criteria. The highest ranked configuration can then be used when constructing the QCA circuit.

An important ranking criterion is defect tolerance. Ranking of different tile configurations within an equivalent-set is performed using the following criteria. The first criterion is based on the undefined outputs. The desired configuration should have a low probability of generating undefined outputs. The second criterion is motivated by defect tolerance. The desired configuration should have a high probability of maintaining the correct output logic function in the presence of defects. Two types of defects are considered, namely the missing cell defect and the additional cell defect. A configuration can be represented as a vector $V_{x_1,x_2\ldots x_9}$, where x_i is 1 when cell C_i is deposited and 0 when cell C_i is undeposited. For example, a configuration where cell C_1 and C_4 are undeposited is represented by 011011111. The best ranked configurations for every function of the orthogonal tile are presented in Table 7.11. The first column in Table 7.11 are all the possible output functions (equivalent-sets). The second column lists the best ranking tile configuration assuming at most a single defect (missing or additional cell) is present. The third column list the best ranking tile configuration assuming single as well as multiple cell defects are present (by exhaustively consider all possible defect patterns). The best ranked configurations can then be used in constructing a QCA circuit which would be also defect tolerant. Details of tile configuration ranking and results on the other tile types are presented in [10].

7.5 LOGIC ANALYSIS

Prior to deposition, the arrangement by which an NFP grid and an assignment of input/output cells are utilized, can significantly change the logic behavior of a

Table 7.11
Best Ranked Tile for Each Output Function for Orthogonal Tile

	Best Configuration	
F	**Single**	**Exhaustive**
A	011011100	011011100
A'	111100100	111100100
B	100111100	100111100
B'	101011000	101011000
C	100011011	100011011
C'	110001101	110001101
Maj(A, B, C)	111111111	111111111
Maj(A, B, C')	011111101	011111101
Maj(A', B, C)	001111111	001111111
Maj(A', B, C')	101101101	100101100
ABC'+AB'C+ A'BC+A'B'C'	111001111	111001111

tile, thus significantly affecting the layout of the final design. However, simulation results have shown that only the orthogonal tile is active, while the remaining four tiles (as well as the interconnection tiles) are passive. Let I denote the number of input cells and V the number of literals found in the largest minterm of the SOP representation of the output function of a QCA circuit exclusive of the undetermined literal (Figure 7.16). As example, for F = Maj(A,B,C) = $AB + AC + BC$, $I = 3$ and $V = 2$. In the analysis below, it is assumed that the input signals are not generated through fixed-polarity cells. The following Lemma characterizes the logic behavior of QCA combinational circuits.

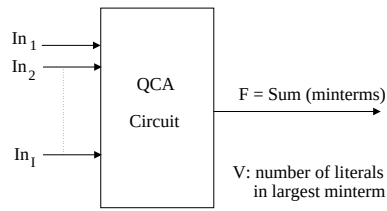


Figure 7.16 QCA Combinational Circuit (Single Output) with I Inputs

Lemma 1: An output combinational function with $V = 2$ cannot be generated by a QCA circuit of $I = 2$.

Proof: This will be proved by contradiction. Consider a QCA circuit S with two input cells I_1 and I_2 and an output function F, hence $F = f(I_1 I_2)$. Two cases can be distinguished: (1) There is no additional cell in the QCA circuit except the two cells with inputs I_1 and I_2. Then, the output is determined by the Coulombic interactions among these cells as determined by the switching induced by a cell on the other cell; in this case, due to adiabatic switching, the output will follow the polarization of the stronger cell, that is, $F = I_1$ (provided I_1 is the stronger cell). (2) Consider next the scenario that there is at least an additional cell (denoted by C) with no input other than I_1 and I_2; C basically acts as a center cell and provides the only input to the remaining cells in S. C interacts with both input cells, however its polarity due to adiabatic switching will be determined by the stronger input cell (say I_1 with no loss of generality). Hence depending on the position of I_1 with respect to C, C will be have a polarization equal to I_1 or the complement of I_1 (i.e. I_1' if the two cells are at a 45^o angle). As C is the only input to the other cells in S, then all these remaining cells cannot be generating a minterm of two literals. In both cases, F has a minterm of only one literal (i.e., $F = f(I_1)$), thus contradicting the initial assumption and proving the Lemma.

The following theorem directly follows from Lemma 1 and the basic operation of an MV.

Theorem 1: The generation of an output combinational function with $V = 2$ requires a QCA circuit with at least $I = 3$.

Simulation results (such as shown in Section 7.2) have shown that Theorem (1) can be extended to the general case of V and I, i.e., the generation of an output function with $V = k$ requires a QCA circuit with $I = k + 1$. At this moment, the proof of this statement remains open due to the inability to find a formal characterization to the problem due to the exponential number of combinations in the arrangements of the input cells. The authors believe that this problem will very likely fall in the NP hard domain.

Among the five considered tiles, the orthogonal tile presents unique processing features, because three MV-like functions can be generated by undepositing multiple cells (albeit the baseline tile implements wire crossing).

- There is no MV-like function with inversion at B. This is caused by the strong polarization of the cell aligned with the center cell of the MV.
- As at least a single inversion can occur internally to an orthogonal tile by undepositing cell(s), then B can be used as a control input and different

functions can be generated. For $B = 0$, $F = \text{Maj}(A', B, C) = A'C$ and $F = \text{Maj}(A', B, C') = A'B'$. This last case corresponds to a 2-input NAND gate. Equivalently, a 2-input NOR gate can be generated by using $B = 1$ and $\text{Maj}(A', B, C')$.

The above considerations show the flexibility of logic functions that can be generated by the orthogonal tile as an active tile. As for generating logic functions, it is obvious that for an NFP grid with a smaller number of cells, some outputs may have an undetermined value, thus making the tile unusable for design. However, in many cases an NFP grid results in a configuration of high polarity.

Next, consider the issue of logic equivalence among tiles, i.e., arrange the input and output cells such that the logic behavior of different tiles can be compared. Due to the large number of output functions and their combinatorial analysis only the cases of having an NFP grid with 8 cells (one undeposited cell) will be considered. The following scenarios are analyzed.

- *Baseline and double fan-out tiles:* if the input A of the baseline is made equal to the other input (i.e., B) and the values of the two outputs ($F1$ and $F2$) are analyzed, then these two tiles show equal behavior in both outputs.
- *Baseline and fan-in tiles:* in the case, only the horizontal outputs are considered and compared (i.e., $F2$ in the baseline tile is ignored). In all cases except when undepositing either cell 5 or 7, the outputs have equal values. This seems to suggests that the second output of the baseline tile accounts for significant interaction by allowing the vertical input to propagate to the horizontal output in these two cases.
- *Double fan-out and fan-in tiles:* only the horizontal output is considered (while connecting together the two inputs of the fan-in tile); both tiles exhibit the same output values except when undepositing cell 5. In this case, complementation occurs in the fan-out tile.
- *Double and triple fan-out tiles:* The upper vertical output of the triple fan-out tile is ignored; for the remaining two outputs it has been verified that these two tiles produce the same values in all cases except when cell 4 is undeposited.

Overall, only partial consistency has been found among tiles; simulation has shown that overall there are instances by which tile behaviors are similar. Especially when a grid is NFP, the 3×3 grid is also influenced by the arrangement of the input and output cells, thus equivalence is very limited.

7.6 EXAMPLES OF QCA CIRCUITS

Different circuit design examples are evaluated in this section. The proposed tile-based design is based on the method described in Section 7.1 using the five types of tiles (based on the 3×3 grids) analyzed in Section 7.3. The tile-based design is compared with SQUARES methodology and gate-based design. Two figures of merit are reported: (1) the rectangular area (in terms of the number of cells) the design occupies; (2) the number of required clocking zones. As noted earlier, the area occupied in the layout by the 3×3 grid (either FP or NFP) is the same as for an INV or MV (with no input/output cells). It is assumed that the design is partitioned into columns of clocking zones and the signals flow from left to right [8]. The following restrictions are applicable to all tiles (the orthogonal tile, the double fan-out tile, the triple fan-out tile and the baseline tile) for timing purposes: (1) signal propagation is from left to right; (2) the outputs of the tile can be only used in a clocking zone to the right of the clocking zone in which the tile is located. When two tiles are placed adjacent to each other, in some cases additional spacing for isolation may be required.

7.6.1 One-bit Full Adder

The one-bit full adder is analyzed first. In this design, different tiles (such as the baseline, double and triple fan-out and the orthogonal tiles) are utilized. The configurations used for these tiles are shown in Figure 7.17. The undeposited cells are denoted by white squares. The deposited cells are denoted by black squares. The baseline tile is used to achieve wire crossing; the double and triple fan-out tiles are used for signal routing; MV as well as MV-like functions are employed using the orthogonal tile. It is interesting that although the MV and the triple fan-out are similar in Figure 7.17, the arrangements in input/output cells cause the two tiles to function differently. Also the flow of signals is enforced by arranging the clocking zones.

The one-bit full adder is built using one MV and two MV-like (with inversion at one of the inputs) gates. The QCA layout as well as the corresponding circuit schematic are shown in Figure 7.18. Three baseline tiles (as wire crossing), two double fan-out tiles, one triple fan-out tile, and three orthogonal tiles (as MV and MV-like) are used in this design. These tiles are connected using passive tiles, which function as wires. The MV gate and the MV-like gates are highlighted by dotted squares. In the design of a full adder, no additional isolation is needed between tiles.

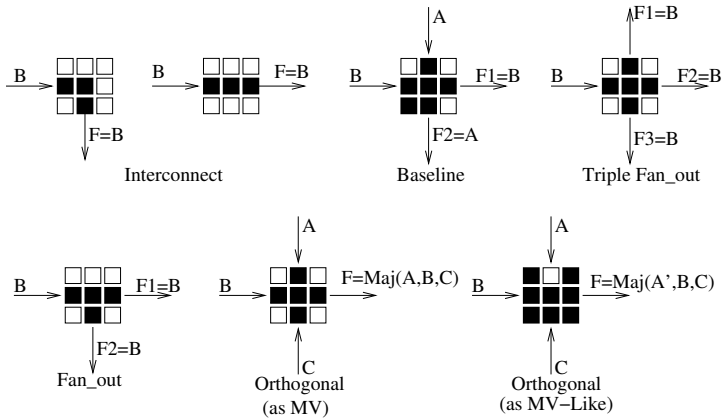


Figure 7.17 Tiles Used in the Design of the Full Adder

The tile-based design uses the same logic schematic as the gate-based design of [11]. The QCA layout of the gate-based design [11] is shown in Figure 7.19. Three MVs and one INV gate are used in this design; it occupies an area of $18 \times 22 = 396$ cells. In the proposed tile-based design, since inversion can be realized using MV-like tiles, no INV is used. Therefore it requires $8 \times 8 = 64$ tiles (an area corresponding to $64 \times 9 = 576$ cells).

The tile-based design can also be compared with the design obtained by SQUARES (shown in Figure 7.20). The tile-based design saves considerable cell area at a significantly reduced latency (in terms of clocking zones). Specifically, the full adder implemented by SQUARES requires $8 \times 7 = 56$ tiles with $56 \times 25 = 1400$ cells. Hence, the full adder using the 3×3 grid as part of the tile achieves a 58% area reduction. Additionally, the proposed method has a smaller delay compared to the SQUARES-based design. Only 8 clocking zones are needed in the tiled-based design, while 15 clocking zones are used in the SQUARES-based design, which corresponds to a 45% reduction in input-output latency.

7.6.2 Parity Checker

A 4-bit parity checker is considered as a second example; this circuit is constructed by using three NXOR gates (with logic "1" at one of the inputs). The QCA layout as well as the corresponding circuit schematic are shown in Figure 7.21. Two double fan-out and three orthogonal tiles (as NXOR) are used in this design. These tiles are

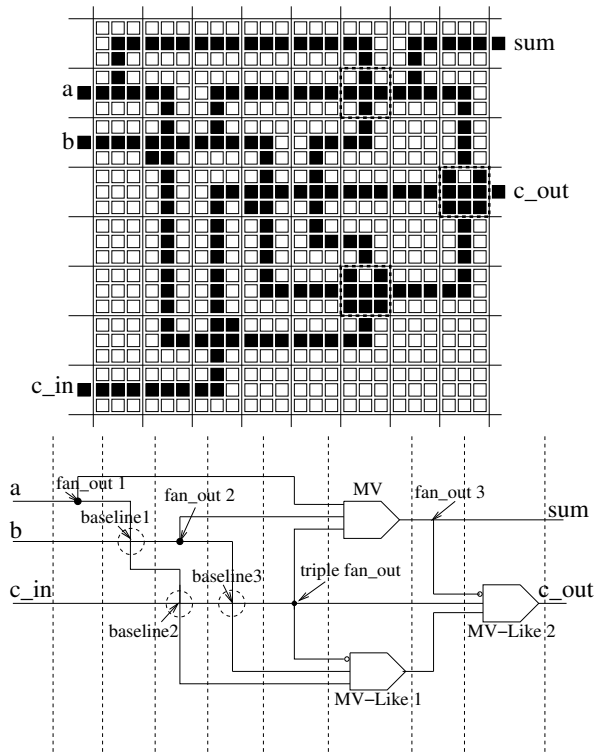


Figure 7.18 Full Adder Using Proposed Tile-based Design

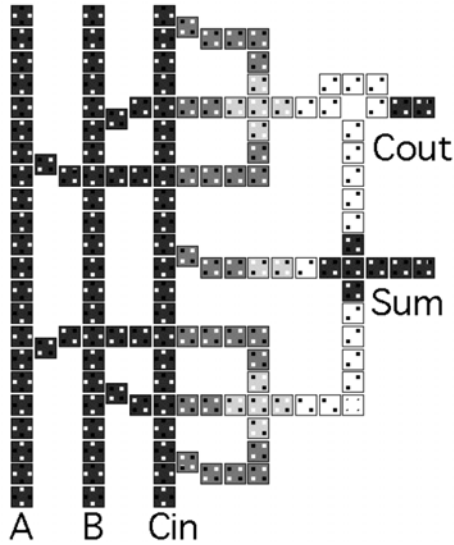


Figure 7.19 Gate-based Design of the Full Adder [11]

connected using interconnection tiles, which function as wires. The NXOR gates are highlighted by dotted squares. As in the case of the full adder, no additional isolation is needed.

As NXOR can be realized using orthogonal tiles, no AND, OR, and INV gates are used. Therefore the design using the proposed tile-based approach requires $5 \times 5 = 25$ tiles (corresponds to an area of $25 \times 9 = 225$ cells). A gate-based design requires an area of $51 \times 29 = 1479$ cells (Figure 7.22); therefore, a tile-based design results in a significant area reduction (68%).

The SQUARES-based design is shown in Figure 7.23. The parity checker implemented by SQUARES requires $8 \times 7 = 56$ tiles (corresponds to an area of $56 \times 25 = 1400$ cells). Hence, the parity checker using the 3×3 grid and corresponding tiles achieves a 58% area reduction. Again the tile-based design has proven to have a reduced delay. A 74% reduction in the number of clocking zones (5 versus 19) is achieved compared with the SQUARES-based design.

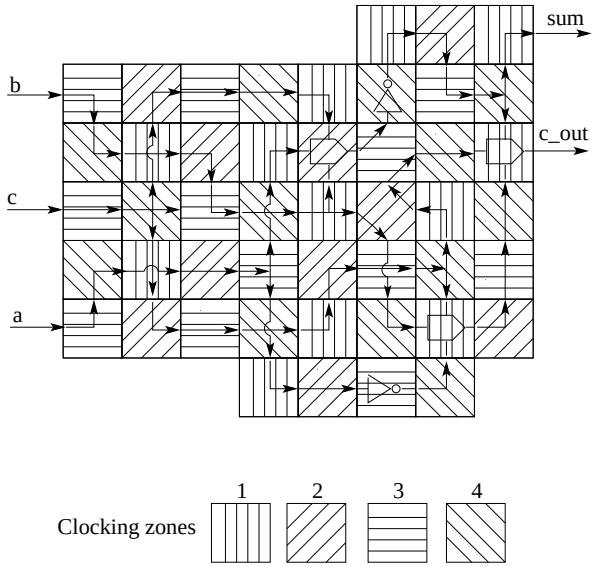


Figure 7.20 Full Adder Using SQUARES-based Design

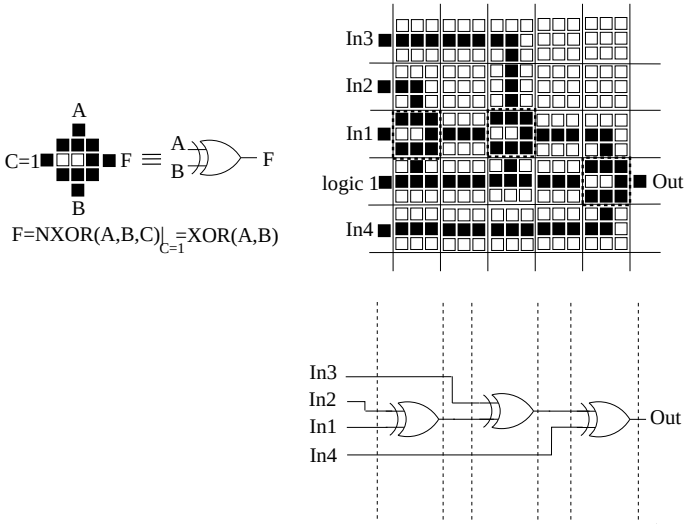


Figure 7.21 4-bit Parity Checker Using Proposed Tile-based Design

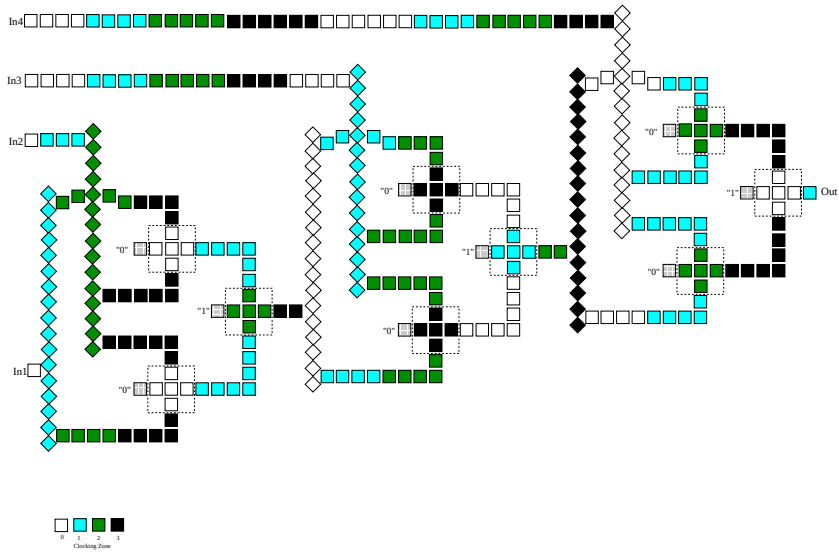


Figure 7.22 Gate-based 4-bit Parity Checker

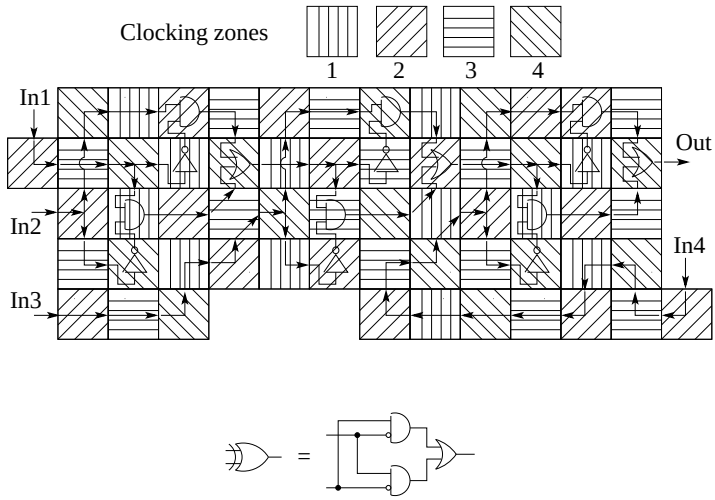


Figure 7.23 4-bit Parity Checker Design Using SQUARES-based Approach

7.6.3 2-to-4 Decoder

A 2-to-4 decoder is built using three MVs and three MV-like (with inversion at one of the inputs) gates. The QCA layout as well as the corresponding circuit schematic are shown in Figure 7.24. Three baseline tiles (as wire crossing), seven double fan-out tiles, six orthogonal tiles (as MV and MV-like) are used in this design. These tiles are connected using interconnection tiles, which function as wires. The MV gate and the MV-like gates are highlighted by dotted squares. Additional spacing for isolating orthogonal tiles from baseline tiles, and the fixed inputs of the orthogonal tiles from wires, is required.

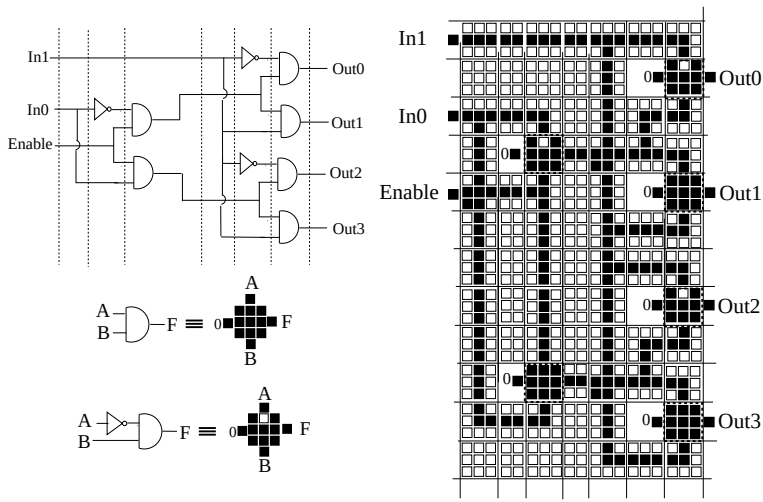


Figure 7.24 Tile-based 2-to-4 Decoder

This design requires $12 \times 5 = 60$ tiles and $12 \times 2 \times 2 = 48$ isolation cells (588 cell area). Compared with a gate-based design (Figure 7.25), which occupies an area of 400 cells, a tile-based design results in a 47% overhead. The 2-to-4 decoder implemented by SQUARES (as shown in Figure 7.26) requires $8 \times 6 = 48$ squares (corresponds to an area of $48 \times 25 = 1200$ cells). Hence, the 2-to-4 decoder using the 3×3 grid and its tiles achieves a 51% area reduction and a 45% reduction in the number of clocking zones (6 versus 11 for the SQUARES-based design).

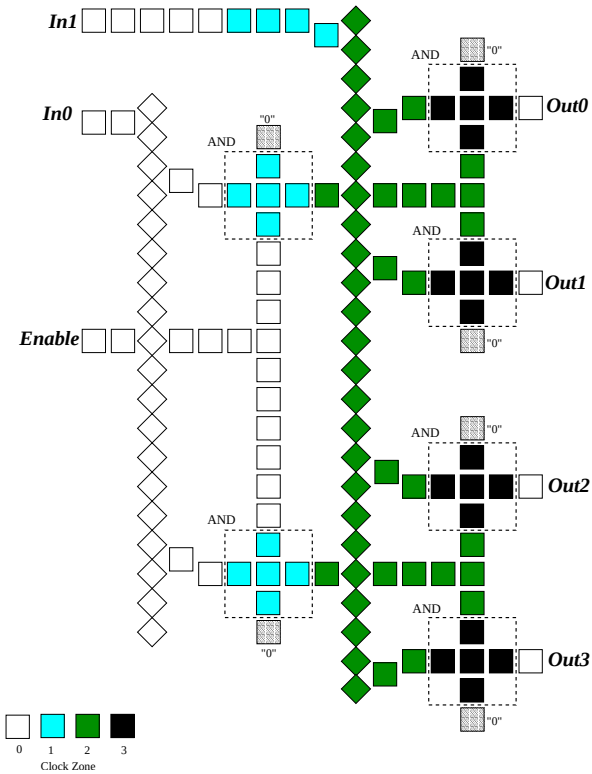


Figure 7.25 Gate-based 2-to-4 Decoder

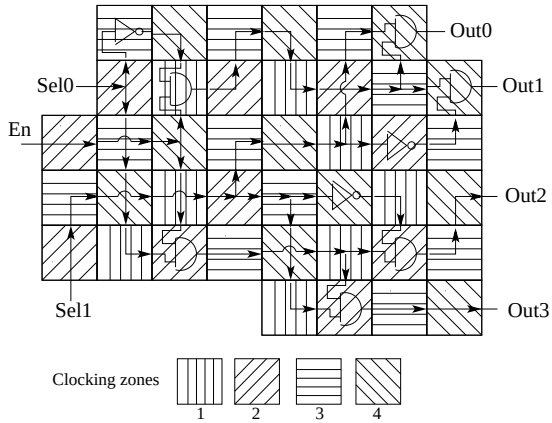


Figure 7.26 2-to-4 Decoder Using SQUARES-based Approach

7.6.4 2-to-1 MUX

The 2-to-1 Multiplexer (MUX) is built with two MVs and one MV-like (with inversion at one of the inputs) gate. The tile-based QCA layout as well as the circuit schematic are shown in Figure 7.27. This design requires 17 tiles, a total of $17 \times 9 = 153$ cells in area. The gate-based design is shown in Figure 7.28 and the SQUARES design is shown in Figure 7.29. The MUX implemented by a gate-based design consists of one INV and three MV gates and it occupies an area of $13 \times 18 = 234$. The tile-based design achieves a 34.6% area reduction. The SQUARES design needs 12 squares, therefore a total of $12 \times 25 = 625$ cells in area. SQUARES has a significantly higher area overhead compared to the tile-based design. The delay for both the gate-based design and SQUARES is 5, while for the tile-based design it is 4.

Table 7.12 summarizes the results implementing the analyzed circuits using the proposed tile-based design, SQUARES and the gate-based design (using MV and INV gates). Note that in some cases (such as for the parity checker) a tile-based design requires a smaller number of deposited cells than a gate-based design.

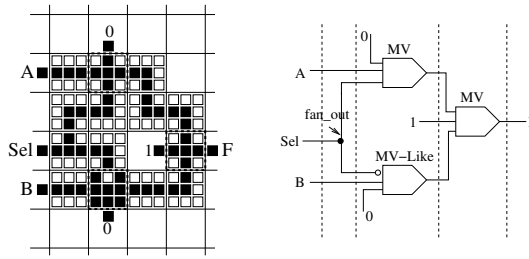


Figure 7.27 Tile-based 2-to-1 MUX

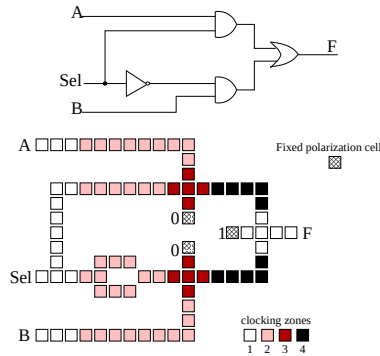


Figure 7.28 Gate-based 2-to-1 MUX

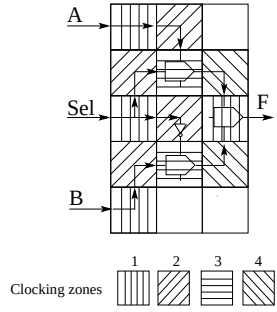


Figure 7.29 SQUARES-based 2-to-1 MUX

Table 7.12
Circuits Using Tiles, SQUARES and Gates

Circuit		2-to-4 Decoder	One Bit Full Adder	Parity Checker	2-to-1 MUX
Tile-based	# of Tiles	60	64	25	17
	Total # of cells	588	576	225	153
	# Clocking zones	6	8	5	4
SQUARES	# of SQUARES	48	56	56	12
	Total # of cells	1200	1400	1400	625
	# Clocking zones	11	15	19	5
Gate-based	Total # of cells	400	396	1479	234
	# Clocking zones	5	5	22	5

7.7 CONCLUSION

In this chapter, the defect tolerance of QCA tiles has been analyzed. The simulation results have shown that PBW by tiling in the presence of undeposited cell defects is still very versatile and robust. The capability of generating the defect-free function is preserved with very high probability for at most one defective cell per tile. Even in the presence of multiple undeposited cells, tiles can still be used in most cases to perform useful logic functions. Throughout the exhaustive simulation (up to 4 defective cells per tile), the following logic functions consistently appear at the output(s), i.e., (1) the wire function, (2) the inverting function, (3) the majority function, and (4) majority-like functions (i.e., as majority function with one or two complemented variables).

This suggests that tile-based design is not as restrictive as using an a-priori device-based configurations in the assembly of QCA circuits. This modularity is reinforced by the flexibility in generating the same set of functions using different tiles with various arrangements for the input/output cells.

This chapter has presented a novel design of combinational circuits by employing basic blocks (referred to as tiles) for assembling QCA circuits prior to cell deposition on a substrate. In this chapter, a tile is a square grid of cells with input/output cells. Grids can be fully populated (FP) or non fully populated (NFP). The tiles are not only versatile in logic implementation but also inherently defect tolerant. With an assignment of input/output cells, different tiles can be utilized for generating a variety of combinational functions. As proposed in this chapter, the basic logic primitive is the MV-like tile; this tile performs the majority function with selective inversion at the input. By combining the functions of MV and INV, the MV-like tile offers an advantage in terms of area efficiency. A set of tiles based on

the 3×3 grids is extensively simulated and analyzed in detail. Logic characterization as well as defect tolerance properties of these tiles are investigated. The presented analysis has confirmed that NFP grids can be efficiently used in designing QCA circuits. Different circuit designs have been presented and compared with SQUARES as well as a traditional QCA gate-based design. It has been shown that a tile-based design achieves considerable area as well as delay (the number of clocking zones between inputs and outputs) reduction compared with SQUARES (and in some cases also compared with a traditional gate-based design). The generation of new combinational functions (such as MV-like functions) and the simple arrangement in the clocking zones make tiles a viable design technique for QCA.

References

[1] Niemier, M. T. and P. M. Kogge, "Problems in designing with QCAs: layout=timing," *International Journal of Circuit Theory and Applications,*Vol. 29, No. 1, 2001, pp. 49-62.

[2] Walus, K., et al., "RAM Design Using Quantum-Dot Cellular Automata," *NanoTechnology Conference,*Vol. 2, 2003, pp. 160-163.

[3] Tahoori, M.B., M. Momenzadeh, J. Huang, F. Lombardi, "Defects and Faults in Quantum-Dot Cellular Automata", VLSI Test Symposium (VTS), 2004, pp. 291-296.

[4] Berzon, D. and T. J. Fountain, "A Memory Design in QCAs Using the SQUARES Formalism," *Proceedings Ninth Great Lakes Symposium on VLSI*, 1999, pp. 166-169.

[5] Orlov, A. O., et al., "Experimental Demonstration of Clocked Single-electron Switching in Quantum-dot Cellular Automata," *Applied Physics Letters*, Vol.77, No. 2, 2000, pp. 295-297.

[6] Jiao, J., et al., "Building Blocking for the Molecular Expression of QCA, Isolation and Characterization of a Covalently Bounded Square Array of two Ferrocenium and Two Ferrocene Complexes," *Journal of the Am. Chem. Society (JACS Communications)*, Vol. 125, No. 25, 2003, pp. 7522-7523.

[7] Qi, H., et al., "Molecular Quantum Cellular Automata Cells: Electric Field Driven Switching of a Silicon Surface Bound Array of Vertically Oriented Two-Dot Molecular QCA," *Journal of the Am. Chem. Society, (JACS Articles),* Vol. 125, No. 49, pp.15250-15259, 2003.

[8] Antonelli, D. A., et al., "Quantum-Dot Cellular Automata (QCA) Circuit Partitioning: Problem Modeling and Solutions ," *Design Automation Conference (DAC)*, 2004, pp. 363-368.

[9] McCluskey, E., *Logic Design Principles*, Englewood Cliffs, NJ: Prentice Hall, 1986.

[10] Vankamamidi, V. and F. Lombardi, "Profiling Tiles for QCA Circuit Design and Defect Tolerance", Internal Report, ECE Dept, Northeastern University, available upon request, 2006.

[11] Zhang, R., et al., "A Method of Majority Logic Reduction for Quantum Cellular Automata," *IEEE Trnsactions on Nanotechnology,* vol 3, No. 4, 2004, pp. 443-450.

Chapter 8

Sequential Circuit Design in QCA

J. Huang, M. Momenzadeh and F. Lombardi

Combinational QCA circuit design has been introduced previously in Chapter 4. Sequential QCA design is investigated in this chapter. The design and characterization of sequential circuits in QCA has not been fully addressed in the technical literature. While sequential elements can be implemented using QCA memory cells, such an approach would be prohibitive in terms of hardware (due to its extensive control circuitry) and very slow in performance. As QCA memories rely on the paradigm of memory-in-motion [1][2], a longer access time should be expected due to the latency in the storage elements. Moreover, sequentiality in QCA does not have the same requirements as in CMOS-based circuits. As indicated in Chapter 3, latching is implicitly implemented in QCA as sequential behavior is dependent on adiabatic switching and the layout of the QCA cells. Adiabatic switching allows one to introduce timing by dividing the QCA cells in zones; this unique feature substantially affects sequential circuits because for example, feedback paths and storage elements could be present in different locations of the layout. In QCA, sequential behavior must be strictly controlled as feedback paths traverses different zones may cause uneven delay among signals.

In this chapter, a detailed analysis of sequential QCA design, which encompasses flip-flop devices as well as circuits, is pursued. Initially, a novel RS-type flip-flop amenable to a QCA implementation is proposed. This flip-flop extends the threshold-based configuration of [3] to QCA by taking into account the timing issues associated with the adiabatic switching of the technology. Defect tolerance property of the RS flip-flop is then presented using a defect model by which single extra and missing cells are considered. The D-type flip-flop, which in QCA is simply

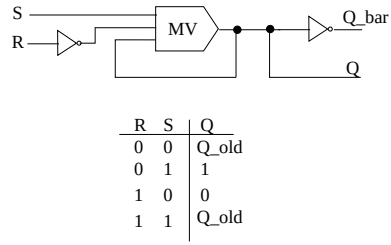
a clocked embedded binary wire, is also considered. Next, unique timing constraints in QCA sequential logic design are identified and investigated. An algorithm for assigning appropriate clocking zones to a QCA sequential circuit is proposed. A technique referred to as stretching is used in the algorithm to ensure timing and delay matching. This algorithm relies on a topological sorting and enumeration step to consistently traverse only once the edges of the graph representation of the QCA sequential circuit. Examples of QCA sequential circuits are provided. Additionally, defect tolerance property of QCA sequential circuits are analyzed.

The analysis hereby presented considers QCA cells to be grown over a Cartesian plane. In a molecular implementation, three dimensional (3D or volumetric) growth is possible [4]. The proposed analysis can be extended also to the 3D case with no loss of correctness. The defect model used here is a single missing or additional cell defect. Through simulation, a single defect is injected in each device; subsequently each of these defects can then be mapped to logic-level faults in the operation of the QCA devices and circuits. All simulations have been performed using QCADesigner v.1.4.0 with its coherence vector engine and a QCA square cell with $2.5nm$ dot size and $10 \times 10nm^2$ cell size are assumed. These values were selected in accordance with scaling features as applicable to QCA technology (refer to Section 5.2.6).

8.1 RS FLIP-FLOP AND D FLIP-FLOP IN QCA

The QCA RS FF is shown in Figure 8.1 and represents a novel QCA extension of the original scheme given in [3] for threshold logic. The basic component in the RS FF is the MV. If the setting input *S* is logic 1 and the resetting input *R* is logic 0, then the stored value of the FF is logic 1. The output value is changed to logic 0 if *R* is logic 1 and *S* is logic 0. When both *S* and *R* have the same value, then the output value remains unchanged. Figure 8.2 shows the QCA layout of the RS FF. The threshold based scheme of [3] requires a three-phase synchronization process; although it is possible to use three-phase synchronization in QCA, the four-phase clocking scheme commonly employed (in QCA a clock cycle requires four clocking zones) is used here. In this design, the number of phases for synchronization is limited by the inner loop in the RS FF. The delay of the inner loop must be a multiple of a full clock cycle, that is, the number of clocking zones in the inner loop must be a multiple of four. In this case, the old value of Q can be made available during the next computation, i.e., after k full clock cycles (where k is an integer). In the RS FF of Figure 8.2, the x and y coordinates are used to identify the QCA cells in the

Cartesian layout. The inner loop of the FF has a delay of one clock cycle; therefore at the output, Q is available 7 clocking zones after R and S have been applied.



R	S	Q
0	0	Q_old
0	1	1
1	0	0
1	1	Q_old

Figure 8.1 Schematic Diagram of the QCA RS Flip-flop

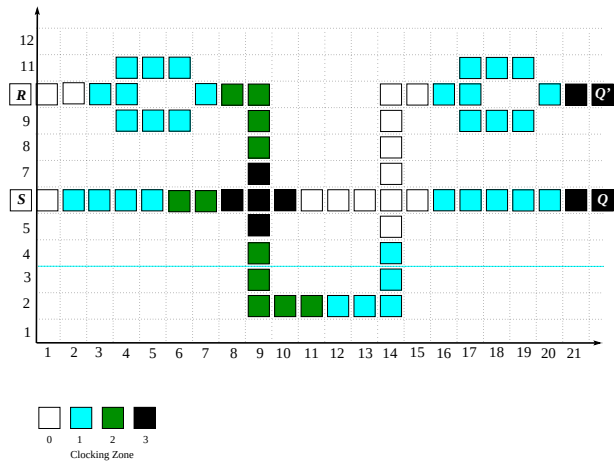


Figure 8.2 QCA Layout of the RS Flip-flop

If adiabatic switching is employed latching is effectively accomplished through timing by using a four phase clocking arrangement. Therefore, a device with an equivalent behavior as a D-type flip flop (D FF) can be constructed by a QCA binary wire with four clocking zones (i.e., it can be buried in a design). In this case, the input signal is delivered to the output after at least one complete clock cycle delay and control is accomplished by timing. The relative simplicity of a D FF over the RS type (no active device required in the former arrangement)

seems to suggest that sequential design in QCA could be achieved at ease within the Cartesian layout. However as shown in the following sections, timing and signal delay must be carefully considered.

8.1.1 Defect Characterization of RS Flip-flop

Single missing and extra cell defects are analyzed in this section for the QCA RS FF of the previous section. The layout of Figure 8.2 is used The radius of effect in these simulations is $40nm$. Figure 8.3 shows the fault free simulation results.

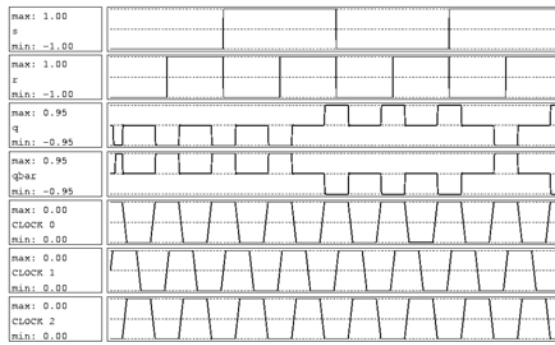


Figure 8.3 QCA RS Flip-flop (Fault-free Case)

For detecting the effects of a defect, a test sequence is utilized; at logic level, this sequence detects stuck-at faults (s-at-1, s-at-0) and up/down transition faults ($\uparrow, \downarrow$). The test pattern for the RS flip-flop is shown in Table 8.1. As the initial value of the RS FF (Q_0) is not known, the input value $RS = 01$ is utilized for setting Q to 1. The vectors $RS = 00$ and $RS = 11$ test for a $\downarrow$ transition fault. Note that $RS = 00$ (or $RS = 11$) can also detect the s-at-1 fault of the output. The vector $RS = 00$ tests for a s-at-1 fault at the R input because this fault is not detectable by the first test vector $RS = 01$ (Q could be 1). With $Q_n = 1$, $RS = 10$ detects s-at-1 and $RS = 00$ and $RS = 11$ test for any $\uparrow$ transition fault. Therefore, if the previous tests ($RS = 00$ or $RS = 11$ with $Q_n = 1$) result in a s-at-1 fault at the output, the next tests ($RS = 00$ and $RS = 11$) will detect these stuck-at faults. Finally, RS is given by 01 to test for a s-at-0 fault .

As per the assumed model (applicable to molecular implementations [5]), single missing and extra cell defects have been simulated. The RS FF schematic diagram is shown in Figure 8.4, which partitions the circuit into numerous devices.

Table 8.1
Test Sequence for RS Flip-flop

Current state Q_n	Test vector (RS)	Operation
d	01	set
1	00	hold 1
1	11	hold 1
1	10	reset
0	00	hold 0
0	11	hold 0
0	01	set 1
1	d	check next state Q_{n+1}

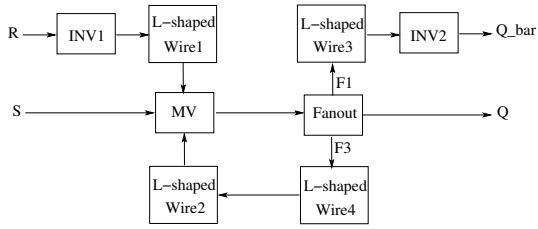


Figure 8.4 Schematic Diagram of Devices in QCA RS Flip-flop

Note the devices includes not only the MV and the INV, but also the L-shape wire. The simulation results (at faulty sites given by the different devices) are shown in Table 8.2 (d denotes the don't care condition). All other single missing cell defects (not reported in this table) result in no faulty output. Single extra cell defects have also been simulated; the results that cause an erroneous output are presented in Table 8.3.

Table 8.2
Simulation Results for RS FF, Single Missing Cell Defect

Faulty Device	Missing cell	Output, Fault-free: $Q_n = d1110001$	Fault
INV1	4,9	d1011011	INV1 behaves as wire
INV1	4,10	d1011011	INV1 behaves as wire
INV1	4,11	d1011011	INV1 behaves as wire
MV	9,5	d1010011	MV as a horizontal wire
MV	9,6	d1010011	MV as a horizontal wire
MV	9,7	d0010010	MV performs Maj(A',B, C')
L-shaped 1	9,10	d0011011	extra INV in L-shaped wire 1
L-shaped 2 or 4	9,2 or 14,2	d1010101	extra INV in L-shaped wire 2/4
L-shaped 3	14,10	d1110001, $Q'_n = Q_n$	extra INV in L-shaped wire 3
Fanout	14,6	d1010101, $Q'_n = Q_n$	extra INV for F1 and F3
INV2	17,9	d1110001, $Q'_n = Q_n$	INV2 behaves as wire
INV2	17,10	d1110001, $Q'_n = Q_n$	INV2 behaves as wire
INV2	17,11	d1110001, $Q'_n = Q_n$	INV2 behaves as wire

Table 8.3
Simulation Results for RS Flip-flop, Single Extra Cell Defect

Faulty Device	Extra Cell	Output, Fault-free: $Q_n = d1110001$	Fault
INV1	6,10	d1011011	INV1 behaves as wire
INV1	7,9	d1011011	INV1 behaves as wire
INV1	7,11	d1011011	INV1 behaves as wire
INV2	19,10	d1110001, $Q'_n = Q_n$	INV2 behaves as wire
INV2	20,9	d1110001, $Q'_n = Q_n$	INV2 behaves as wire
INV2	20,11	d1110001, $Q'_n = Q_n$	INV2 behaves as wire

8.2 TIMING CONSTRAINTS IN QCA SEQUENTIAL DESIGN

The proposed FF represents the basic device by which sequential designs can be built in QCA. In conventional logic design, synchronous operation is usually implemented in a sequential circuit. This circuit can be represented by a Mealy machine that consists of two parts: the flip-flops and the combinational logic. This model is applicable to QCA as well. However, in QCA the clock signal controls not only the FFs, but also the combinational gates. The entire QCA circuit is pipelined and latched by the clock signals. An important timing constraint in a QCA design is that for every logic gate, all inputs must arrive at the same time (all inputs in the same clocking zone). Further, in synchronous sequential logic, all flip-flops should compute at the same time. Therefore it is necessary to ensure that all paths from the outputs of the flip-flops (passing through the combinational logic) to the inputs of the flip-flops have the same delay (i.e., the number of clocking zones), thus *enforcing* the condition that signals arrive at the inputs of the flip-flops at the *same time* (strict matching).

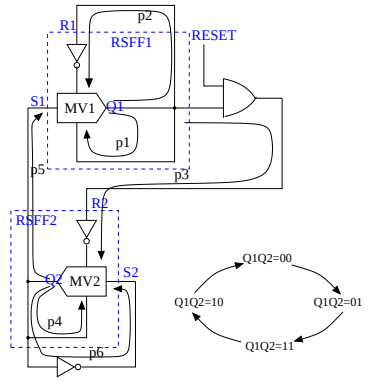


Figure 8.5 QCA 2-Bit Grey Code Counter

8.2.1 Timing Constraints Using RS Flip-flops

For QCA sequential designs with RS FFs, the timing constraints are as follows: (1) All state variables must be updated at the same time, as required by a synchronous sequential design. If the state variables are chosen to be the output of the MVs in the RS FFs, then all MVs in the RS FFs must be in the same clocking zone. (2) For each MV, all inputs must arrive at the same time, that is, all paths from the output of an MV in the RS FF to the input of an MV in the RS FF must have the same delay (as given by the number of clocking zones).

This timing constraint is illustrated in the example shown in Figure 8.5; this circuit is a 2-bit Grey code counter. Two RS FFs are employed in the design whose state transition diagram is shown in Figure 8.5. For the two flip-flops to compute at the same time, $MV1$ and $MV2$ must be placed in the same clocking zone (i.e., $Q1$ and $Q2$ are in the same clocking zone). The timing constraint that must be applicable for correct sequentiality consists of ensuring that for each MV, all three inputs must arrive at the same time. This corresponds to the condition by which the paths $p1$,$p2$,$p3$,$p4$,$p5$ and $p6$ must have the same delay (given by the number of clocking zones). Two of these paths ($p1$ and $p2$) are the inner (feedback) loops of the RS FF. As the inner loop must have a delay that is a multiple of the clock cycle (one clock cycle consists of four clocking zones), then a timing arrangement must be implemented in the QCA design.

As explained in Chapter 3 the number of QCA cells that can be placed in the same clocking zone is bounded. In a complex sequential design, a path that

goes through a large number of combinational logic gates may require more than one complete clock cycle. In this case, all the paths must be "stretched" to match the delay of the longest path. If all paths have a delay of exactly k cycles, then a valid output will be produced every k cycles. However, k should have a small value (preferably 1) to maintain the data flow in the pipeline of the QCA circuit.

8.2.2 Timing Constraints using D Flip-flops

Similar timing constraints are applicable to the QCA design of sequential circuits using D FFs. In a D FF-based design, the D FF is effectively "buried" in other logic gates. An example is shown in Figure 8.6, this sequential circuit is the so-called *traffic light*. The pair $Q1Q2$ defines the state variables of the traffic light as follows: $Q1Q2 = 00$ is green, $Q1Q2 = 01$ is yellow and $Q1Q2 = 11$ is red. W is the pedestrian crossing (input) signal. $W = 1$ denotes a pedestrian's request to cross. The circuit functions are shown in the state transition diagram in Figure 8.6. It can be seen from the corresponding QCA layout that the first D FF is "buried" inside the loop $p1$, while the second D FF is "buried" inside the loop $p2$. The state variables $Q1, Q2$ can be chosen anywhere in the cells of the loops. In this example, they are chosen to be the output at the fan-out points ($f1$ and $f2$) as shown in Figure 8.6. Thus, $f1$ and $f2$ must be in the same clocking zone such that the two state variables are updated simultaneously. It is required that the paths $p1$,$p2$,$p3$,$p4$ and $p5$ must have the same delay. The longest delay path is incurred in $p3$ and $p4$, both of which have a delay of two clock cycles. Therefore, the other paths must be adjusted (for matching a two clock cycle delay); this process is referred to as *stretching* and it will be described in detail next.

8.3 ALGORITHM FOR CLOCKING ZONE ASSIGNMENT

8.3.1 Algorithm Outline

In this section, a clocking zone assignment algorithm to meet the timing constraints in QCA sequential circuits is proposed. This algorithm is a novel modification of the algorithm introduced in [6] for satisfying the timing constraints in reconvergent paths for QCA circuits. In this algorithm, the gates consist of MVs, INVs, fan-outs and wires. For example, a wire gate is basically a binary wire that performs the identity function. It is assumed that each gate is placed in a clocking zone.

The algorithm assigns clocking zones to each gate by *enumerating* them. The proposed method can be explained as follows. Each gate (active gate, fan-out, wire)

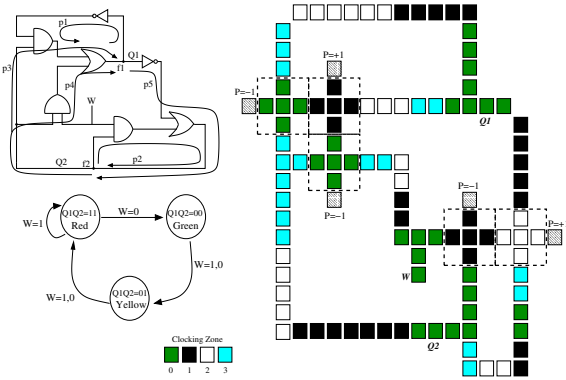


Figure 8.6 QCA Traffic Light

is initially assumed to be in its own clocking zone. Hence, a *directed cyclic graph* $G' = (V', E')$ can be used to model the sequential circuit. Each gate is represented by a vertex in G'. If the output of gate u drives the input of gate v, a directed edge $(u, v) \in E'$. Next, G' is transformed into a *directed acyclic graph (DAG)* G by breaking (opening) the feedback loops. Then, a vertex (as starting point in the execution of the algorithm) referred to as the *Super Source*, is added to the graph representation of the QCA circuit. The preliminary step of this process consists of adding edges between the Super Source and the vertices that represent the flip-flops. The algorithm takes this transformed graph as input and finds for each vertex of the DAG the longest path from the Super Source. Since the graph is a DAG, the vertices can be arranged in a *topologically sorted order*. The Bellman-Ford algorithm can then be applied in this order to find the longest paths [7]. The topological sorting step ensures that a vertex is processed only when all its parents have been processed. In the algorithm, each node is given a label for its clock number (this corresponds to the number of zones from the Super Source). The clock number of a child node is one more than the largest clock number of its parents.

Next, the paths are stretched (by adding edges) for delay matching. In the stretching process, if the number of clocking zones of two nodes with a common edge differs by i, then $i - 1$ vertices are added between these nodes. When adding vertices, the algorithm considers the effect of shared paths to reduce the number of additional clocking zones. Figure 8.7(a) illustrates an example. Assume that clocking zones U_1, U_2 and U_3 are required between node A and nodes B and C,

respectively. Using the proposed algorithm for stretching (Figure 8.7(b)), node U_1 is added to the shared path and only two clocking zones are inserted.

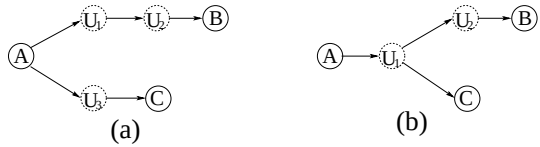


Figure 8.7 Stretching Considerations for Path Sharing

Let the *center gate* of a FF be defined as the gate (MV, INV, fanout) whose output is the state variable. The state variables in a synchronous sequential design must be updated at the same time, therefore all center gates must be in the same clocking zone. The center gate can be chosen arbitrarily inside an FF, for convenience the center gate for an RS FF is chosen to be the MV inside the FF itself (Figure 8.1). For a DFF, the center gate is the gate whose output is the state variable. The center gate can be chosen anywhere in the feedback loop. For example the fanouts $f1$ and $f2$ in the traffic light example are the center gates (Figure 8.6). From the previous discussion, the following conditions must be met at circuit-level to meet the timing constraints:

- All center gates must be in the same clocking zone.
- All paths from an output of a center gate to an input of a center gate must have the same delay.
- All paths from the primary inputs to a center gate must have the same delay.

Synchronous sequential circuit can be modelled by the Mealy machine, as shown in Figure 8.8. Timing constraints must also be imposed on any combinational logic prior to the Mealy machine itself. This input logic block is shown in Figure 8.8 as CL1. Therefore, another timing requirement must be considered such that all primary inputs are in the same clocking zone as the FF. In this arrangement, the primary inputs are synchronized with the state variables of the sequential machine. This is achieved by adding an edge from the Super Source to each primary input.

8.3.2 Algorithm Detail

A new *graph-based model* is proposed for the QCA sequential circuit. This is formally given by an unweighted directed cyclic graph $G' = (V', E')$. This

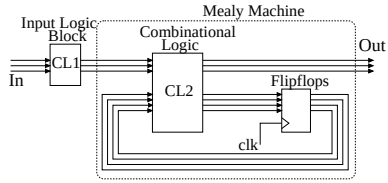


Figure 8.8 Mealy Model of Sequential Machine

graph is transformed into a directed acyclic graph $G = (V, E)$ with the so-called *vertex splitting* step as follows. The definition of center gate has been introduced previously in Section 8.3.1.

- Each center gate CG is represented by two vertices $u' \in V$ and $u'' \in V$ (vertex splitting).
- All inputs to each CG are modeled by edges entering u'
- All outputs of CG are modeled by edges leaving u''.

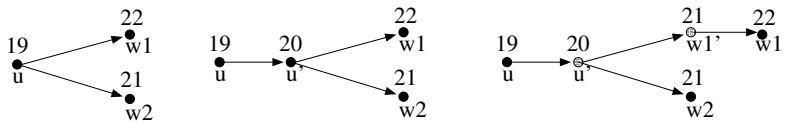
u' is called an *input center vertex*, while u'' is called an *output center vertex*. There are two types of loop inside a graph in the proposed QCA sequential circuit model: (1) self loops (from a flip-flop to itself), (2) connecting loops (from one flip-flop to another). The process of splitting the center gates cuts both these types of loops, thus breaking all feedback paths in the circuit. Consider Figure 8.8; if all flip-flops in a sequential circuit are open-looped, then the resulting circuit is combinational. Therefore, the corresponding graph is a DAG because no loop exists in a combinational circuit by definition (no feedback path).

In the next step, a Super Source vertex (denoted by ss) is added to G. An edge is added from ss to each of the output center vertices as well as the primary input vertex. After this modification, G is still a DAG. Let the clocking zone of gate u be denoted as $clk(u)$. The proposed algorithm (denoted as AssignClk(), see Algorithm 1) assigns the clocking zones to each gate $u \in V$. The algorithm starts at ss and initially assigns $clk(ss) = -1$. A sorting step is done for the DAG such that the vertices are arranged in a topologically sorted order. Next Function NumerateDAG() (see Algorithm 2) is executed such that each vertex u is assigned a $clk(u)$. In NumerateDAG() the vertices are processed in a topologically sorted order, such that u is processed only after all its parents have been processed. After the execution of Function NumerateDAG(), the clock assignment satisfies the following condition: let the parents of vertex u be $v1, v2....vn$, then $clk(u)$ is assigned the maximum

value of $clk(v1), clk(v2)....clk(vn)$ plus 1. As all center output vertices as well as the primary input vertices are children of ss, then they will be assigned to clocking zone 0.

After executing the Function NumerateDAG(), further processing is needed for the center vertices. For each center vertex CG, the two vertices u'' and u' represent the same gate; therefore, they must be in the same clocking zone. This is clocking zone 0 because $clk(u'') = 0$ for all output center vertices u''. So the first requirement is that the clocking zone of all input center vertices must satisfy the condition $clk(u')\ modulo\ 4 = 0$. However, this is not necessarily applicable after the execution of the Function NumerateDAG(). Hence an adjustment may be required. The second requirement is that all input center vertices must have the same clock number. Let k' be the maximum of $clk(u')$ among all input center vertices. Let k be the smallest integer that is greater than k' and is a multiple of 4. k is assigned to $clk(u')$ for all input center vertices u'. For example, if k' is 6, then k is set to 8.

After all vertices have been numerated, the algorithm AssignClk() calls Function StretchPath() (see Algorithm 3) and stretches the short paths to match the longer ones as follows. The function $minchild(v)$ finds the minimum clk among v's children, where $children(v)$ denotes the set of v's children. Initially, the path between the input center vertices u' and its parent v is stretched. If $clk(u') > clk(v) + 1$, then node u' is extended, such that a number of nodes (given by $clk(u') - clk(v) - 1$) are added between v and u'. Next, stretching is performed on all other vertices u. Stretching of a common path is considered first. Let w be the child of u with the smallest clk value among all u's children. Node u is extended such that a number of nodes (given by $clk(w) - ckl(u) - 1$) are added between u and the children of u. Finally, stretching of non-common paths between u and its children is performed. This is illustrated by the example as shown Figure 8.9, where u has two children, $w1$ and $w2$. Stretching is required because $clk(u) = 19$, $clk(w1) = 22$ and $clk(w2) = 21$. Initially common path stretching is performed (in which u' is added to the graph). Then non-common path stretching is performed, in which $w1'$ is added.



(a) before stretching (b) after common stretching (c) after non–common stretching

Figure 8.9 Example of Stretching of Common and Non-common Paths

The final graph that satisfies all timing requirements has the following characteristics: (1) For every center gate CG, $clk(u'') = 0$, $clk(u') = k$ where $kmodulo4 = 0$; (2) For every primary input vertex u, $clk(u) = 0$. (3) For each edge (u, v) from u to v, $clk(u) + 1 = clk(v)$. The algorithm and corresponding subroutine are given in Algorithms 1 and 2 and 3. As four-phased clocking is used in QCA, the clocking zone of any gate u must be $clk(u)\ modulo\ 4$. For example if $clk(u) = 7$, u should be in zone 3.

The complexity of the proposed algorithm is computed as follows. The topological sorting step of G can be performed in $O(|V| + |E|)$=$O(|E|)$ time [7]. Function NumerateDAG() has a complexity of $O(|E|)$, because each edge is traversed exactly once. Stretching considers each edge in G once by inserting the required vertices and edges. Since for each original edge, at most $O(|V|)$ vertices need to be added, then stretching has a complexity of $O(|V||E|)$. Thus, Function AssignClk() has a time complexity of $O(|V||E|)$.

8.3.3 Algorithm for Coplanar Device

In this section, an assignment algorithm is introduced to satisfy the timing constraints imposed by the so-called coplanar device in QCA circuits. In this device, two separate QCA wires cross on the Cartesian plane without affecting each other. This operational feature is valid provided the wires are in the same clocking zone at the crossing point. Therefore, the values of the clocking zones of the wires at the crossing point must be $modulo\ 4$ of each other.

The proposed algorithm can be explained as follows. The crossover is identified in the graph by a pair of vertices (denoted as $co*$ and $co+$). An example is shown in Figure 8.10; a is the parent of $co*$, while c is the parent of $co+$; b is the child of $co*$, while d is the child of $co+$. No cycle is introduced because the crossover is treated as two separate vertices (hence, with the crossover device the graph G is still a DAG). The algorithm introduced in the previous section is applicable to crossover vertices with a slight modification to Function NumerateDAG(). The topological sort as well as the stretching steps in Algorithm AssignClk() can be used for the crossover vertices with no modification by treating each crossover as two separate vertices. However in Function NumerateDAG() $co*$ and $co+$ must be in the same clocking zone, i.e., $(clk(co^*) - clk(co^+))\ mod\ 4 = 0$. During the execution of Function NumerateDAG(), if a crossover vertex is encountered, Function Crossover() (or Algorithm 4) is called. When reaching a crossover vertex for the first time (say $co*$), the execution of the algorithm is unchanged and *clk* is assigned to the crossover vertex ($co*$) . When the crossover vertex is accessed for

the second time (i.e., $co+$), then $co+$ is numbered such that it is placed in the same clocking zone as $co*$. For example, in Figure 8.10 assume $co*$ is reached first. Since $clk(a) = 13$, so $clk(co*) = 14$. The crossover point is then processed for a second time when $co+$ is reached. As $clk(co*)\ mod\ 4 = 2$, then $clk(co+)$ must satisfy $clk(co+)\ mod\ 4 = 2$. Assume $clk(c) = 19$, then $clk(co+)$ is assigned 22, which is the smallest integer i bigger than 19 for which $i\ mod\ 4 = 2$.

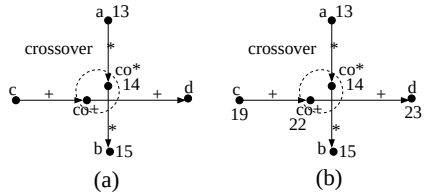


Figure 8.10 Examples of the Coplanar Device

8.3.4 Examples of QCA Circuits

The first example is the traffic light discussed previously, the graph model of which is shown in Figure 8.11. The center gates are the fanout gates $f1$ and $f2$. The original graph is given in Figure 8.11(a). It can be seen the center gates are the fanout gates. The graph after applying Function NumerateDAG() is shown in Figure 8.11(b). As $f1' = 6$, $f2' = 4$, then in the next step the algorithm assigns $clk(f1') = clk(f2') = 8$ as multiple of four. Two additional wires are added between the parent of $f1'$ (an OR gate) and $f1'$, while four additional wires are added between the parent of $f2'$ (a wire gate) and $f2'$. The final graph after stretching is shown in Figure 8.11(c); this matches the layout of the traffic light given previously in Figure 8.6. Note that in the layout the state variables are Q1 and Q2, both in clocking zone 0.

The next example is the 2-bit Grey code counter discussed previously in Figure 8.5. The original graph is given in Figure 8.12(a); the center gates are the majority voters $f1$ and $f2$. The graph after applying the Function NumerateDAG is shown in Figure 8.12(b), in which clock numbers 6 and 7 are ultimately assigned to $f1'$ and $f2'$ respectively. In the next step the algorithm assigns $f1' = f2' = 8$ and additional wire gates are added. The final graph is shown in Figure 8.12(c), and the corresponding layout is given in Figure 8.13. In the layout MV $f1$ is in clocking zone 2 while MV $f2$ is in clocking zone 3. This is due to stretching to ensure that

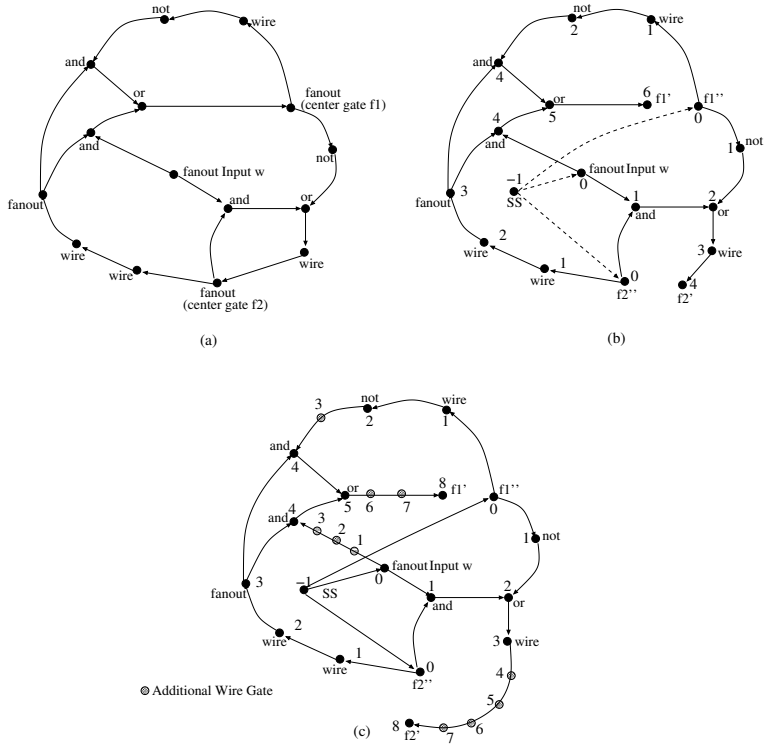


Figure 8.11 Graph Model of the Traffic Light

the state variables $Q1$ and $Q2$ are in the same clocking zone. In this example $Q1$ and $Q2$ are in clocking zone 0.

Another example is the QCA circuit for S27 from the ISCAS89 sequential benchmark set, which includes the coplanar crossing. The QCA layout is shown in Figure 8.14. The schematic is shown in Figure 8.15. A QCA implementation requires 3 D flip-flops, 11 active gates (2 inverters, 1 AND, 2 NANDs, 2 ORs, 4 NORs). The original graph is given in Figure 8.15(a). The center gates are f_1, f_2, f_3. The graph after applying NumerateDAG() is shown in Figure 8.15(b). In the next step the algorithm assigns $f1' = f2' = f3' = 26$. The stretching part is not shown for simplicity.

8.4 DEFECT CHARACTERIZATION OF QCA SEQUENTIAL CIRCUITS

In this section, the defect characterization of sequential circuits designed using QCA FFs are presented; two examples are provided.

A sequential circuit in QCA relies on the paradigm of memory-in-motion through devices such as the RS-type flip-flop of the previous section. Memory-in-motion employs QCA wires to store information over multiple clocking zones in the QCA layout. For memory-in-motion, data is *circulated* in a loop, so the same defect may have *different implications* on the operation of a QCA circuit. In the presence of a cell defect (as assumed in this chapter), the behavior of a QCA sequential circuit must be analyzed together with its timing features: clocking through the zones basically achieves *latching* of logic values. Hence, in contrast with combinational circuits an extra or missing cell defect can result in an erroneous signal (due to a new functionality of a device) to be propagated only after a delay. This means that robustness of the QCA circuit is a function of the layout and its timing organization. Two examples of sequential circuits in QCA are analyzed next.

Semaphore: The first circuit is the so-called *semaphore* as commonly used for resource access. The schematic diagram and the state transition diagram of the semaphore are shown in Figure 8.16. w is the input signal, $w = 0$ denotes a request for resource access, while $w = 1$ denotes no request. $Q = 0$ denotes a granted access, while $Q = 1$ denotes that the resource is being accessed. This circuit operates as follows: (1) when the resource is released ($Q = 0$), and there is a request for access, the request will be granted; next, the resource is accessed ($Q = 1$). If no request is present, the resource remains in the non-accessed (or released) state; (2) when the resource is accessed ($Q = 1$), next the resource is released ($Q = 0$) to wait for the next request. The corresponding QCA layout is shown in Figure 8.17;

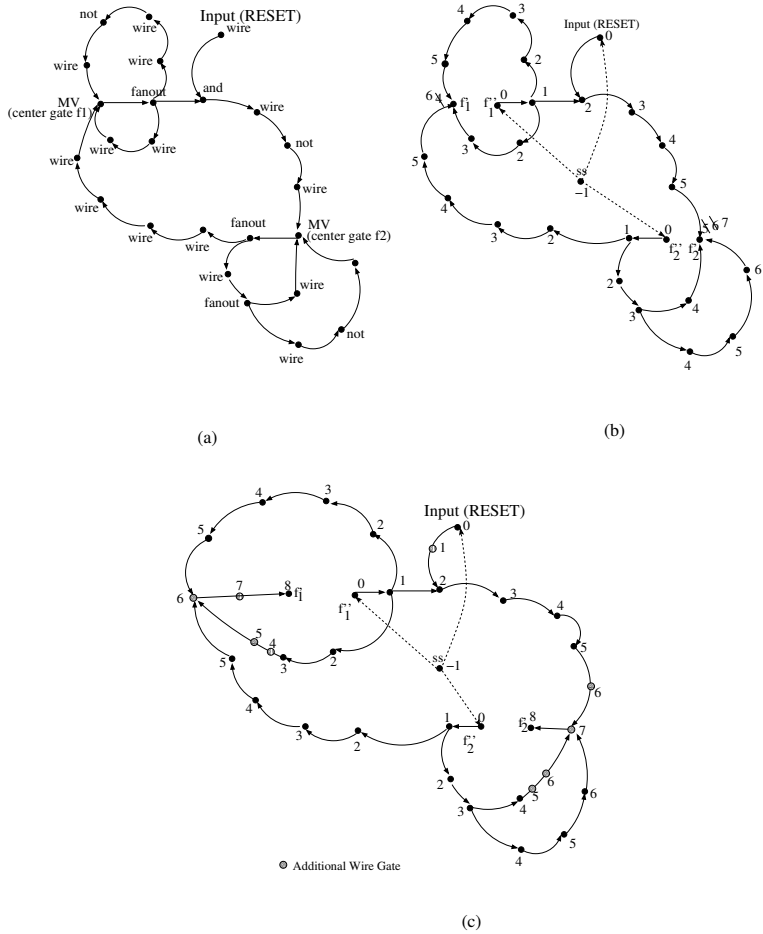


Figure 8.12 Graph Model of 2 Bit Grey Code Counter

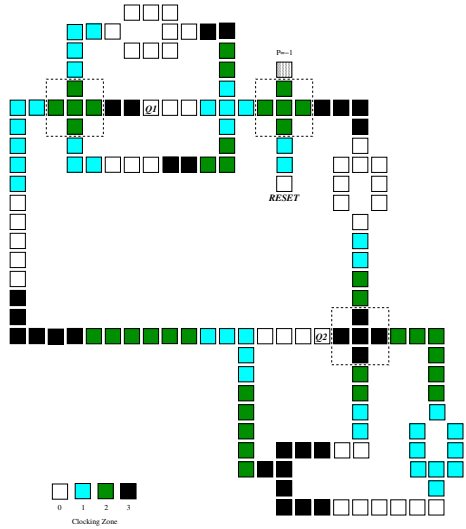


Figure 8.13 Layout of 2 Bit Grey Counter

the active devices (MVs) are highlighted by dotted squares. In this layout, $p3$ is the longest path and its delay consists of two clock cycles. As per the timing constraint discussed previously, paths $p1$ and $p2$ are stretched to two clock cycles such that the three paths will have the same delay (strict matching).

In this particular case, stretching is not strictly required; as the top input of the MV in the RS flip-flop is Q and the bottom input is Q' (where Q' is the logic inverse of Q), then due to the voting nature of the MV, the top and bottom inputs complement each other. Therefore, the output of the MV follows the horizontal input and as long as $p1$ and $p2$ have the same delay, the circuit will function correctly. The MV basically operates as a wire, so it is possible to remove $p1$ and $p2$ by replacing the MV in the RS FF with a QCA wire without changing the functionality of the circuit. The resulting QCA layout is shown in Figure 8.18(a); the circuit consists of a QCA loop with one INV and one OR gate. Simulation has confirmed that this circuit operates as desired. Moreover, this layout can be further simplified to operate within a single full clock cycle, i.e., a valid output is produced every clock cycle. This new circuit is shown in Figure 8.18(b). This circuit effectively shows that in QCA, a binary wire traversing zones in one full clock cycle (four clocking zones) behaves as a D FF. This is the *simplest instance*

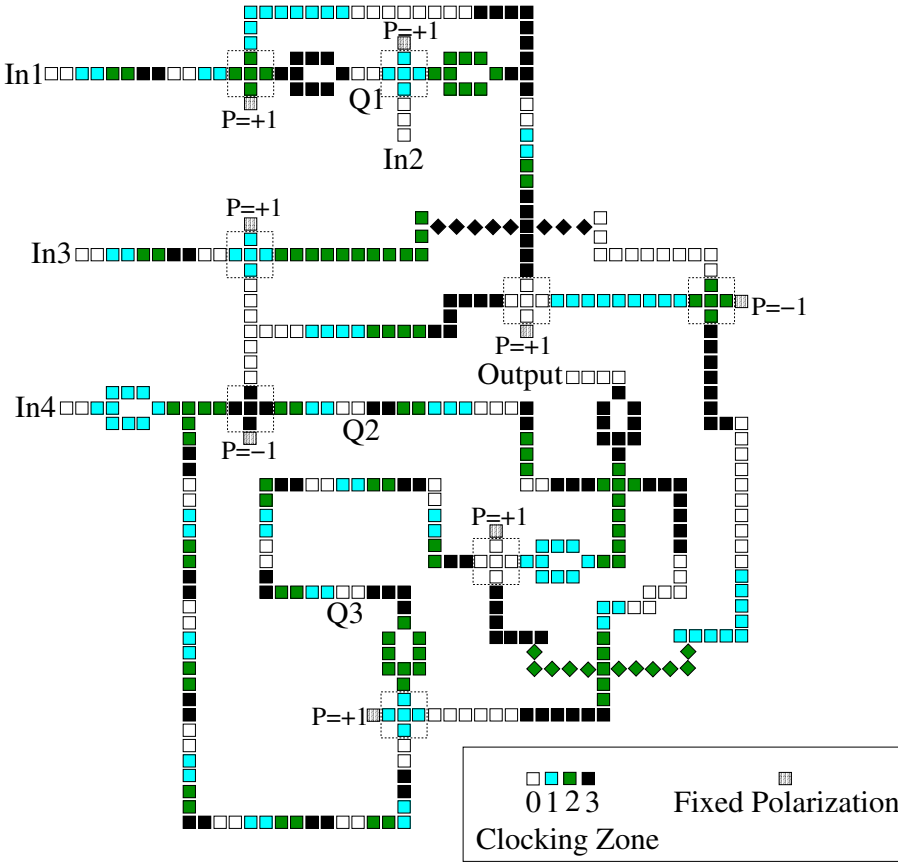


Figure 8.14 QCA Layout of ISCAS89 S27 Benchmark

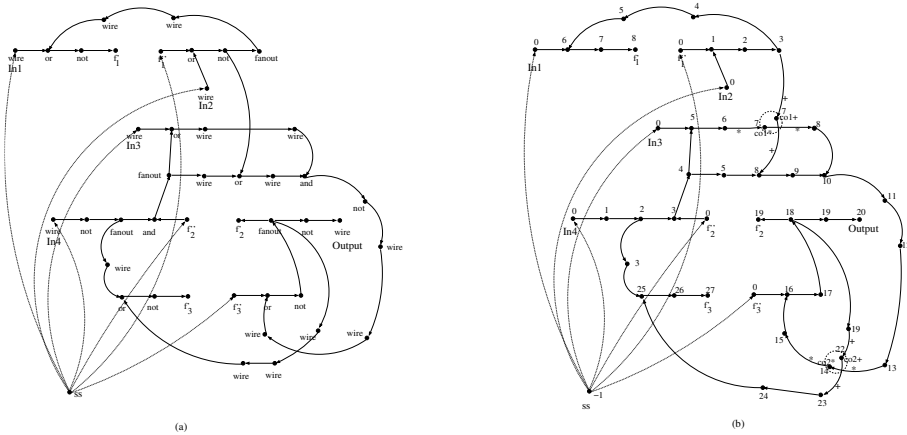


Figure 8.15 Graph Model of QCA ISCAS89 S27 Benchmark

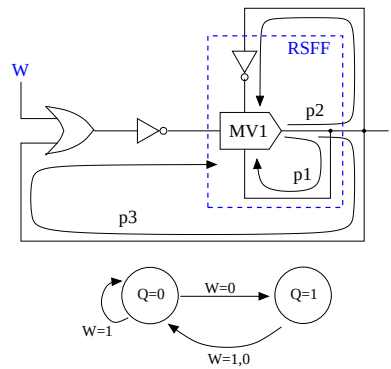


Figure 8.16 Schematic Diagram of the QCA Semaphore

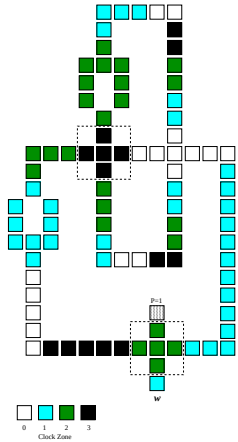


Figure 8.17 QCA Layout of the Semaphore

of processing-by-wire as D is propagated to Q every clock cycle. Hence, the design shown in Figure 8.18 is also an example of designing a sequential circuit using a QCA D FF.

Single missing as well as extra cell defects have been simulated. For the semaphore shown in Figure 8.18, defects were injected using the x and y coordinates of the cell layout. The results are given in Table 8.4. The test sequence that has been applied for fault detection in the presence of a single defect is given by $W = 110000011$. The first vector resets the circuit to $Q = 0$ using $W = 11$. Then, the sequence $W = 00000$ is used to test the circuit's operation to toggle between $Q = 0$ and $Q = 1$. Finally, $W = 11$ resets the circuit to $Q = 0$. Note that defects due to missing cells (3,8) (6,6) (7,4) (5,4) (4,4) (2,5) and (2,7) and extra cells (3,7) (5,7) (5,6) (5,5) (4,3) result in a fault free output.

Lock: As a second example, consider the QCA implementation of the so-called *lock*. This circuit effectively toggles between two states until it remains in a locked position (as dependent on the input signal $w = 1$). The initial state is defined by the signal $Reset = 0$. The schematic and the state transition diagrams are shown in Figure 8.19. One D-type flip-flop, two AND gates and one OR gate are used; the QCA layout is shown in Figure 8.20. The simulation results are given in Figure 8.21, in which the valid output is generated every two cycles.

Single missing/extra cell defects have also been considered for the lock. The simulation results are shown in Table 8.5 (all cell defects not reported in this table

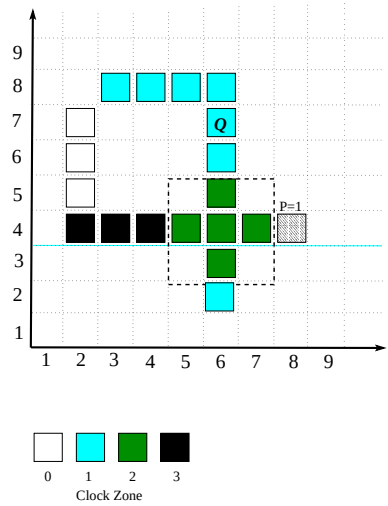


Figure 8.18 Simplified QCA Layout of the Semaphore

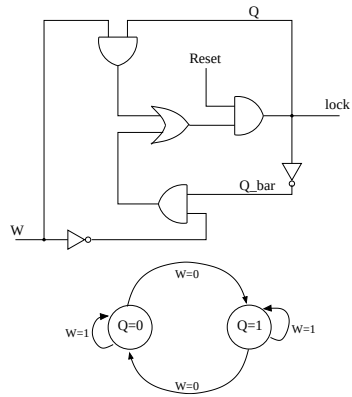


Figure 8.19 Schematic Device and State Diagrams of the QCA Lock

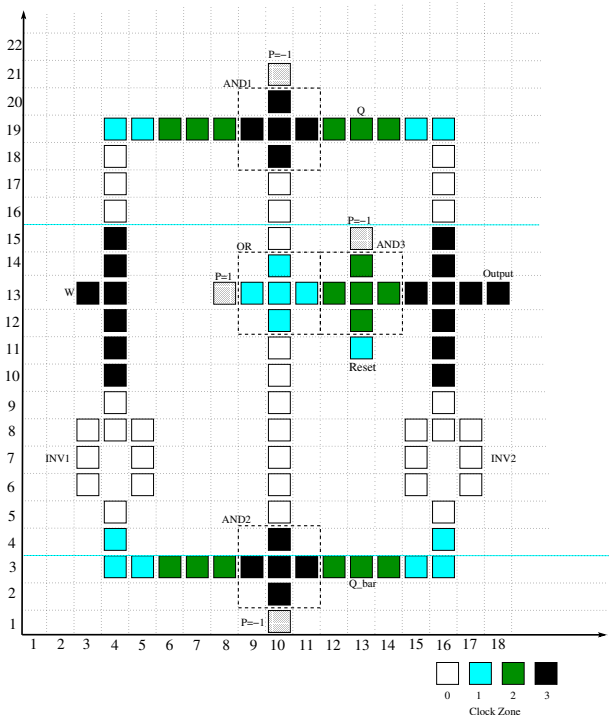


Figure 8.20 QCA Layout of the Lock

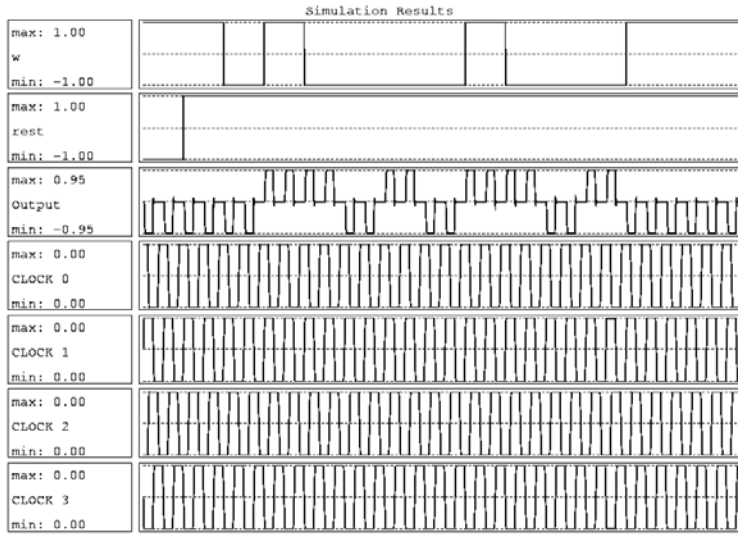


Figure 8.21 Simulation Results of the QCA Lock

Table 8.4
Simulation Results for Single Missing and Extra Cell Defect in Semaphore

Input W=1100,0001,1; Fault free output Q=0010,1010,0		
Missing Cell	**Output Q**	**Comment**
4,8	0000,0000,0	Signal unable to propagate through the gap
5,8	0000,0000,0	Signal unable to propagate through the gap
6,8	1111,1111,1	Extra INV in the loop (Q s-at-1)
6,5	0000,0000,0	MV as a wire for the horizontal output (fixed 1)
6,4	1000,0011,1	
6,3	0000,0000,0	MV as a wire for the horizontal output (fixed 1)
3,4	1000,0011,1	
2,4	1111,1111,1	Extra INV in the loop (Q s-at-1)
2,6	0000,0000,0	Q s-at-0
Extra Cell	**Output Q**	**Comment**
2,8	1111,1111,1	Missing INV in the loop

result in a fault free output). The test sequence is given by $Reset$=0011,1111,1111,11 and $W = 1111, 0011, 0000, 0000, 11$. Note that since the circuit has a delay of two clock cycles, a valid input is applied every two cycles (and therefore a valid output is also observed every two cycles). First the test sequence resets the circuit to $Q = 0$ by $Reset = 00$. Next, $W = 11$ forces the circuit to lock on $Q = 0$. Then, $W = 00$ causes the circuit to toggle to $Q = 1$. The circuit is then locked on $Q = 1$ when $W = 11$. Therefore, $W = 0000, 0000$ is used to toggle the circuit between $Q = 0$ and $Q = 1$. Finally, $W = 11$ locks the circuit to $Q = 1$. The fault free output sequence is given by $Q = 0000, 1111, 0011, 0011, 11$.

8.5 DISCUSSION AND CONCLUSION

The analysis and results presented in the previous sections point to interesting features for the design and defect tolerance of QCA sequential circuits in molecular implementations. Sequential elements in QCA have unique properties as related to processing-by-wire characteristics: as timing in each feedback path must be closely monitored to synchronize all signals, sequential operation can be obtained through either a modification of the MV (as basic device for constructing a RS-type FF), or a QCA wire (as providing the delay feature of a D-type FF). In both cases, robust sequential design requires *tight synchronization* and *clocking zone adjustments*. A circuit-level characterization of sequentiality in QCA has been pursued. This includes the conditions by which the delay incurred in feedback paths of cells in the Cartesian plane can be taken into account, such that correct operation with respect to timing and delay can be achieved. An algorithm that modifies these features using a technique referred to as stretching, has been proposed. This algorithm relies on a topological sorting and enumeration step to consistently traverse only once the edges of the graph representation of the QCA sequential circuit. Timing considerations are accounted by matching the delays along all paths by inserting QCA wires. Unique QCA devices (such as the coplanar crossing) have been considered. Examples of QCA sequential circuits have been described. Finally it should be noted that the proposed algorithm does not guarantee to be optimal for inserting the wire gates. Also the logic-level design rather than the physical layout has been considered in this algorithm. Therefore if the resulting final graph can not be drawn into the QCA layout, further stretching may be required to match the delays.

The unique features of sequential design in QCA are also evident for defect tolerance: using a molecular-based model that includes a single extra or missing cell, simulation results have shown that in both QCA devices and circuits, these defects are mostly evidenced at logic-level by extra inversion and MV malfunction (i.e. MV behaves like a wire) faults. For the proposed RS flip-flop, the INVs are the most defect sensitive devices to single extra cells; for single missing cell defects, the MV confirms its sensitivity on the strongest input (i.e., the center input signal B) while L-shaped wire interconnects show an erroneous inversion (due to a defect occurring in the corner cell). It has been shown that the RS-type FF proposed in this chapter for QCA implementation is robust and can be efficiently used in designing sequential circuits. The simulation presented in the chapter shows that the defect-tolerant operation of a QCA sequential circuit seems to have the same faulty behavior as the flip-flop; this characteristic results in logic faults that change

the functionality of the QCA devices. Overall, this chapter has shown that device-level defective behavior can be extended to circuit-level; consistent results have been obtained under a single cell defect model. Moreover, sequential elements (i.e., the flip-flops) in QCA show the same logic faults which are encountered in basic combinational gates, such as the MV and INV.

Function AssignClk($G(V, E)$)

Data : G as DAG graph of the circuit;
u' and u'' input and output center vertices; ss super source vertex

begin
topological sort $G(V, E)$
$clk(ss) \leftarrow -1$
NumerateDAG(G)
- - assign same clock to all input center vertices
for *all input center vertices* u' **do**
$k \leftarrow$ maximum of $clk(u')$
end
if k modulo 4 $\neq 0$ **then**
$k \leftarrow k + 4 - (k$ modulo 4)
end
for *all input center vertices* u' **do**
$clk(u') \leftarrow k$
end
- - stretch any edge (v, u') where u' is an input center vertex
for *each edge* (v, u') **do**
while $clk(v) + 1 < clk(u')$ **do**
$i \leftarrow clk(u') - clk(v) - 1$
$V \leftarrow V \cup x_i$ - - create new vertex x_i
$clk(x_i) \leftarrow clk(v) + 1$
$E \leftarrow E \cup \{(v, x_i), (x_i, u')\} - \{(v, u')\}$
$v \leftarrow x_i$
end
end
- - stretch other vertices
StretchPath(G)
end

Algorithm 1: *Clock Assignment Algorithm for Sequential QCA Circuits*

```
Function NumerateDAG(G(V, E))
Data    : G as DAG graph of the circuit
          ss super source vertex

begin
    for each u ∈ V do
        clk(u) ← −∞
    end
    for each u taken in topologically sorted order do
        for each of u's child v do
            if clk(v) < clk(u) + 1 then
                clk(v) ← clk(u) + 1
            end
        end
    end
end
```

Algorithm 2: *Algorithm to Numerate Vertices in a DAG*

Function StretchPath($G(V, E)$)

Data : G as DAG graph of the circuit;
u' and u'' input and output center vertices;
ss super source vertex

begin
  for *each* $u \in V$ **do**
    - - stretching the common path between u and u's children
    while $clk(u) + 1 < minchild(u)$ **do**
      $i \leftarrow minchild(u) - clk(u) - 1$
      $V \leftarrow V \cup x_i$ - - create new vertex x_i
      $clk(x_i) \leftarrow clk(u) + 1$
      $E \leftarrow E \cup \{(u, x_i), (x_i, children(u))\} - \{(u, children(u))\}$
      $u \leftarrow x_i$
    end
    - - stretching the non-common path
    for *every child* v *of* u **do**
      while $clk(u) + 1 < clk(v)$ **do**
        $i \leftarrow clk(v) - clk(u) - 1$
        $V \leftarrow V \cup x_i$
        $clk(x_i) \leftarrow clk(u) + 1$
        $E \leftarrow E \cup \{(u, x_i), (x_i, v)\} - \{(u, v)\}$
        $u \leftarrow x_i$
      end
    end
  end
end

Algorithm 3: *Algorithm for Path Stretching*

Function Crossover($G(V, E), co, v$)

Data : G as DAG graph of the circuit;
co crossover vertex;
v current parent vertex of crossover vertex; $passed \in \{False, True\}$;
$a \in \{*, +\}$ is attribute of current edge (v, co)

begin

- - crossover first pass

if $co.passed = False$ **then**

$co.passed = True$

if $a = *$ **then**

$clk(co^*) \leftarrow clk(v) + 1$

end

else

$clk(co^+) \leftarrow clk(v) + 1$

end

end

- - crossover second pass

else

if $a = +$ **then**

x$\leftarrow (clk(co^*) - clk(v) - 1) modulo 4$ where x$= 0, 1, 2, 3$

$clk(co^+) \leftarrow clk(v) + 1 + x$

end

else

x$\leftarrow (clk(co^+) - clk(v) - 1) modulo 4$ where x$= 0, 1, 2, 3$

$clk(co^*) \leftarrow clk(v) + 1 + x$

end

end

end

Algorithm 4: *Crossover Algorithm*

Table 8.5
Single Missing and Extra Cell Defect Results for the Lock

Reset Input=0011,1111,1111,11; W=1111,0011,0000,0000,11 Fault free output=0000,1111,0011,0011,11		
Missing Cell	**Output Q**	**Comment**
4,3	0111,0011,0000,0000,11	extra INV from INV1 to AND1, so $Q = W$
3,8	0111,0011,0000,0000,11	INV1 behaves as wire, so $Q = W$
4,8	0111,0011,0000,0000,11	INV1 behaves as wire so $Q = W$
4,13	0110,1001,0101,0101,10	$W \rightarrow W'$
4,19	0000,1111,1111,1111,11	cell(7,19) s-at-1, so $Q_n = Q_{n-1} + Q'_{n-1}W'$
9,19	0000,1100,1100,1100,00	AND1 behaves as a vertical wire, output s-at-0, so $Q_n = Q'_{n-1}W'$
10,19	0000,1100,1100,1100,00	AND1 behaves as a vertical wire, output s-at-0, so $Q_n = Q'_{n-1}W'$
10,16	0000,1100,1100,1100,00	faulty OR gate, so $Q_n = Q'_{n-1}W'$
10,14	0111,1111,1111,1111,11	OR gate behaves as wire, output s-at-1, so Q s-at-1
10,13	0111,1111,1111,1111,11	OR gate behaves as wire, output s-at-1, so Q s-at-1
10,12	0111,1111,1111,1111,11	OR gate behaves as wire, output s-at-1, so Q s-at-1
13,19	0111,0011,0011,0011,11	AND1 output follows W
16,19	0111,0011,0011,0011,11	cell(12,19) s-at-1, AND1 output follows W
10,2	0000,1111,0011,0011,11	AND2 behaves as a vertical wire
10,4	0000,1111,0011,0011,11	AND2 behaves as a vertical wire
10,3*	0000,0000,0000,0000,00	AND2's output s-at-0
13,14	0101,1010,0110,0110,10	AND2 behaves as a vertical wire
13,12	0101,1010,0110,0110,10	AND2 behaves as a vertical wire
13,13	1111,0000,1100,1100,00	AND2's output=Maj (A',B,C')
15-17,8	0000,0000,0000,0000,00	INV2 behaves as wire
16,13	0110,1001,0101,0101,10	inversion on vertical fanout wires
16,3	0000,0000,0000,0000,00	corner behaves as INV
Extra Cell	**Output Q**	**Comment**
4,6	0111,0011,0000,0000,11	INV1 behaves as wire, so $Q = W$
3,5	0111,0011,0000,0000,11	INV1 behaves as wire, so $Q = W$
5,5	0111,0011,0000,0000,11	INV1 behaves as wire, so $Q = W$
15,5	0000,0000,0000,0000,00	INV2 behaves as wire
17,5	0000,0000,0000,0000,00	INV2 behaves as wire
16,6	0000,0000,0000,0000,00	INV2 behaves as wire

References

[1] Frost, S. E., et al., “Memory in Motion: A Study of Storage Structures in QCA,” *1st Workshop on Non-Silicon Computation*, 2002.

[2] Walus, K., et al., “RAM Design Using Quantum-Dot Cellular Automata,” *NanoTechnology Conference*,Vol. 2, 2003, pp. 160-163.

[3] Muroga, S., *Threshold Logic and Its Applications*, New York, NY: John Wiley and Sons Inc., 1971.

[4] Qi, H., et al., ”Molecular Quantum Cellular Automata Cells: Electric Field Driven Switching of a Silicon Surface Bound Array of Vertically Oriented Two-Dot Molecular QCA,” *Journal of the Am. Chem. Society, (JACS Articles)*, Vol. 125, No. 49, 2003, pp.15250-15259.

[5] Dysart, T. J., P. M. Kogge, C. S. Lent and M. Liu, “An Analysis of Missing Cells Defects in Quantum-Dot Cellular Automata,” *Proc. IEEE NanoArch*, 2005.

[6] Niemier, M. T. and P. M. Kogge, ”Exploring and Exploiting Wire-Level Pipeling in Emerging Technologies,” *Proceedings IEEE International Symposium on Circuits and Systmes (ISCAS)*, 2001, pp. 166-177.

[7] Cormen, T. H., et al., “Introduction to Algorithms”, *McGraw-Hill*, 2001.

Chapter 9

QCA Memory

V. Vankamamidi, M. Ottavi and F. Lombardi

9.1 INTRODUCTION

As introduced in Chapter 3, QCA has many desirable features for processing [1]; for example, clocking and timing can be adjusted as functions of the cells in a Cartesian layout. Low power consumption (power gain has been demonstrated by clocking of the cells), high density and regularity are readily applicable to QCA; therefore, different circuits and systems can be designed using QCA. A system that is well suited to this technology, is the QCA implementation of large memories. However, large memory designs in QCA present unique characteristics due to their architectural structure (such as the tournament bracket in cell placement). Sequential circuit design has been explored previously in Chapter 8, this chapter investigates the design of large-scale memory in QCA.

For storage, QCA utilizes the so-called paradigm of memory-in-motion, i.e. the state of a memory must be kept in movement in the QCA cells. It is possible to distinguish two types of memory architectures: parallel and serial architectures. A parallel architecture offers the substantial advantage of low latency because at each memory cell, only one data bit is stored, so there is no delay in that bit reaching the Read/Write circuitry. In CMOS, Random Access Memory (RAM) is usually designed using parallel architectures (Figure 9.1), in which the Select/Control signal reaches all memory cells (MC) in a row (thus forming a word) during the same clock cycle. This results in an output that is read simultaneously. The one-bit-per-memory cell in QCA reduces latency, but the replication of Read/Write circuitry

Table 9.1
Comparison of Parallel and Serial QCA Memory Architectures

Feature	Parallel Architecture	Serial Architecture
Latency	Low	High
Read/Write Circuitry	Duplicated for every bit	Shared between multiple bits
QCA-cell and Control-cell Count	High	Low
Zone Count and CMOS Circuitry	Complex	Simple
Memory Density	Low	High

for each memory bit increases hardware count (QCA-cell, Control-cell, Clocking-zone). Therefore the parallel architecture provides faster operation of memory at a reduced density.

In a serial memory design, multiple bits are stored in each memory cell and Read/Write circuitry is shared between them. The most obvious advantage is with respect to hardware. And since read/write circuitry for memory cells in QCA is relatively more complex compared to CMOS implementation, sharing it with multiple memory cells simplifies the architecture. Thus serial architecture provides memory at higher density with simpler design although at lower operation speed.

Table 9.1 summarizes the comparison of parallel and serial QCA memory architectures.

In this chapter, novel architectures for both parallel and serial memory architectures are presented for implementation in QCA.

A parallel memory architecture for QCA is first introduced. The parallel architecture utilizes an arrangement in the memory cell design by which storage is achieved by moving data back and forth along a line of QCA cells. This line-based arrangement results in substantial savings in the number of zones and underlying circuitry's complexity for clocking the QCA memory. To obtain this result, the proposed architecture requires two additional clocking signals as the line-based operation of the memory cell needs three zones and a four-step process whose timing is different from the commonly used quasi-adiabatic switching.

Next, a serial memory based on the utilization of basic building blocks referred to as tiles, is proposed. Tiles are used in the memory cell to construct a loop for moving the memory state in different QCA circuits (memory-in-motion)

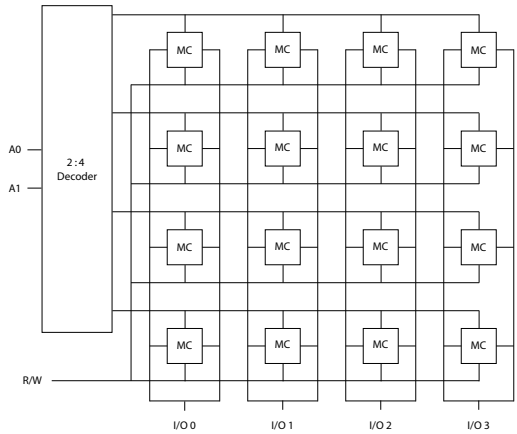


Figure 9.1 Block Diagram of a Two-Dimensional Random Access Memory (RAM)

as well as input/output capabilities for the Read/Write operations. The combination of tile-based design and memory-in-motion by state-looping results in a novel timing/clocking arrangement by which semi-adiabatic switching can be implemented using two additional signals within a two-stage operational cycle. The serial memory proposed in this chapter uses different tiles to allow bidirectional signal propagation. The closed QCA loop which is required to store data, is formed by using a pair of parallel wires connected together at both ends. The resulting rectangular-shaped loop is partitioned into multiple columns of tiles (Figure 9.15). Each tile alternates between one of the two stages in the operational cycle, Hold and Switch; adjacent tiles are always in different stages, so at any given time, half of the tiles are in the Hold stage and the other half are in the Switch stage. When a tile is in the Hold stage, it holds two bits of data, one for each horizontal wire and; when it is in Switch stage, it holds no data.

9.2 REVIEW OF QCA MEMORIES

The design of memories must first consider clocking and timing as important features for QCA operation. The use of a quasi-adiabatic switching technique as commonly employed for QCA circuits requires a four-phased clocking signal that is supplied by CMOS wires buried under the QCA circuitry for modulating the electric field.

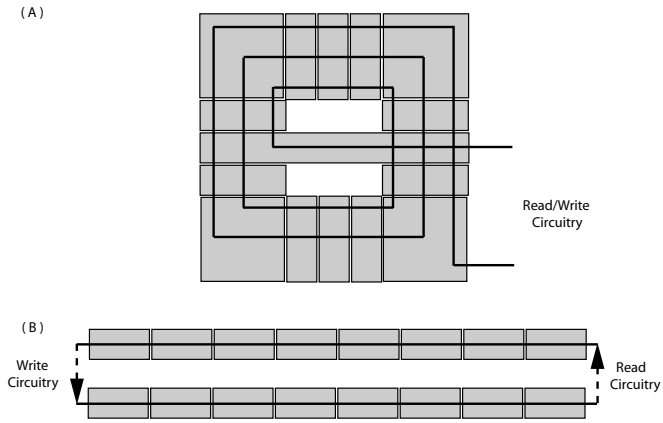


Figure 9.2 Serial Memory Cell Architectures. (A) Memory Spiral Architecture Presented by Frost et al. (B) Memory Loop Architecture Used by Berzon et al.

For quasi-adiabatic operation of a cell, the four phases are Relax, Switch, Hold and Release. During the Relax phase, there is no interdot barrier and a cell remains unpolarized. During the Switch phase, the interdot barrier is slowly raised and a cell attains a definitive polarity under the influence of its neighbors. In the Hold phase, barriers are high and a cell retains its polarity, and finally in the Relax phase, barriers are lowered and a cell loses its polarity. As for timing of QCA circuits, they are partitioned into multiple clocking zones, and all cells in a zone are clocked according to this periodic four-phased signal. Therefore, a straightforward approach to implement a memory by QCA is to maintain a cell (zone) in the Hold phase as long as its value must be retained for storage. The main problem with this rather obvious approach is the requirement of an explicit control of the CMOS clock signal from the decoder (which is implemented in QCA). Also, the transfer of signals from QCA to CMOS requires a complicated sensing process using sophisticated electrometers. For a truly QCA-based implementation, memory must be kept in motion, i.e, the memory state has to be continuously moved through a set of QCA cells connected in a loop partitioned into four clocking zones, and at any given time one of them is in the Hold phase to retain the information.

In [2], an early attempt was made to design a QCA memory using the so-called SQUARES formalism. The basic principle of this technique is to define a set of equally sized blocks, each performing a basic function in QCA (as either logic, or interconnect). These blocks can then be tiled together to design more complex QCA

circuits. The obvious advantage of this technique is the ease in the geometric layout; also, this formalism allows a design to be highly modular. However, as the blocks are of standard size (in SQUARES a 5×5 grid is used), a substantial unutilized area appears in each block, thus causing spatial redundancy and lower density in the overall design. The memory designed using SQUARES is a serial architecture [Figure 9.2 (B)]. Each memory cell is a closed loop QCA wire that is partitioned into multiple clocking zones equal to four times the number of bits stored in the loop. This creates a large number of clocking zones even for a modest memory size, thus requiring a considerable amount of CMOS circuitry to generate the clocking signals. Finally, additional control circuitry (such as comparators) must be utilized to make the memory bit-addressable. This results in a quite high hardware penalty per memory cell.

Researchers at Notre Dame University have introduced the H-Memory architecture [3] whose main objectives are high density and uniform access time. The H-Memory has a complete binary tree structure with control circuitry at each node; as the memory spirals are at the leaf nodes, an integration of logic and memory is accomplished in the layout, but the control circuitry and memory are logically separate (similarly to CMOS design). However unlike conventional designs, control and data bits are serialized. The bit stream enters the memory structure at the root node and traverses down the tree by utilizing one control bit for routing at every node in the path. The architectural choice of dealing with serial bit streams also results in rather complex control logic for QCA. The router at each internal node has ten gates and six feedback loops; each loop requires four clocking zones for its implementation. The circuitry at the leaf nodes (i.e., the memory cells) requires 11 gates per node. Also, the memory cell at each leaf node is a spiral allowing storage of several bits, while sharing clocking zones between multiple concentric loops [Figure 9.2 (A)]. In this design, the memory size at each spiral and the cell count do not have a linear relationship; each outer loop has an increasing diameter, thus requiring more QCA cells for its implementation (although its storage capacity remains constant).

Reference [4] has proposed a conventional parallel memory architecture (such as encountered in CMOS-based RAM design) for QCA, i.e., by storing one bit at each memory cell. The single-bit memory cells allow the design of a simple Read/Write circuitry; each memory cell is implemented using 158 QCA cells and the Select signals are separately generated using decoders. The main disadvantage of this approach is the same as the one encountered in [2]; namely, data in each memory cell is stored using a closed QCA wire loop (which is partitioned into four clocking zones). Therefore, the memory design requires a large number of clocking

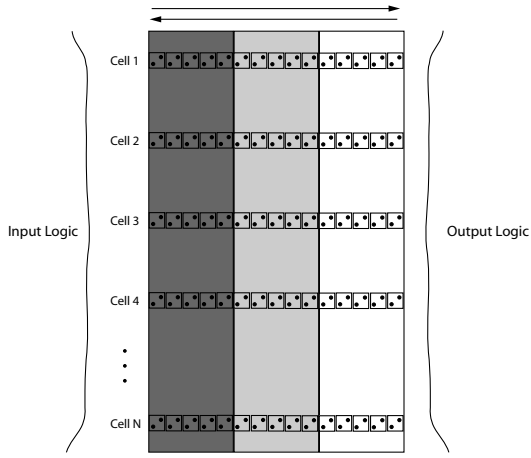


Figure 9.3 Line-based Memory

zones, thus complicating the underlying CMOS circuitry for providing the required clocking signals. Also, since clocking zones cannot be shared between memory loops their dimensions are very small making clocking of such zones difficult if not infeasible.

9.3 PARALLEL MEMORY ARCHITECTURE

9.3.1 Proposed Parallel QCA Memory Design

In this section, a fundamentally different design of a parallel QCA memory is introduced. This architecture is based on a novel logic arrangement for the MV, namely the wires to an MV can behave differently (either as input or output) in time depending on the clock phase in which they are operative. This arrangement combined with a new clocking strategy, overcomes the limitation of a traditional unidirectional flow of logic signals in QCA. The new arrangement of the MV is exploited in the design of a parallel memory architecture for QCA by which the number of clocking zones for implementing the memory is independent of its size. This is accomplished by sharing zones among all memory cells in a column. A further advantage of this approach is a reduction in the CMOS circuitry to provide the clock signals. The hardware requirements for the Read/Write control logic are

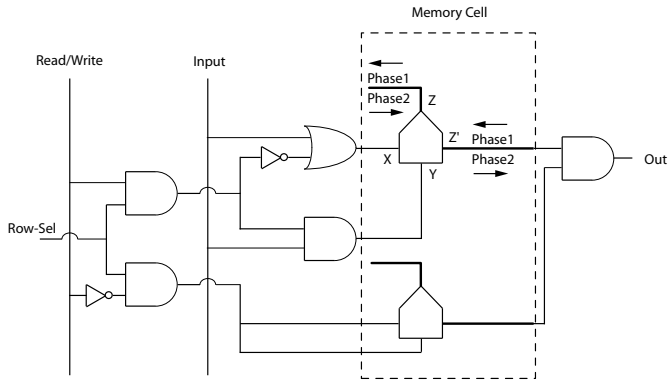


Figure 9.4 QCA Memory Cell with Input and Output Logic Circuitry

comparable to [4]. Also, the Read/Write control logic is very simple compared to other designs in the literature [2] [3]. The only additional cost with respect to [4] is that the design requires two additional clock signals whose Hold/Relax times are different from the original (equally timed four-phased) clock signal. The basic principle of the proposed approach is to store bits by moving them back and forth in straight QCA lines, hence this technique is referred to as line-based. The design of a one-bit memory architecture is shown in Figure 9.3. The proposed line-based QCA memory cell employs three consecutive clocking zones forming a timing row. At any given time, at least one of the three zones is in Hold phase, such that the memory state is retained. Whenever Zone 3 is in the Hold phase, the memory state can be read out based on the values of the Select signals; whenever Zone 1 is in the Switch phase, then a new input state can be written to the memory cell. Multiplexing between the current value and new input value is controlled by the Select signal; this is performed by the MV in Zone 2 and the systematic switching of the clocking zones. Such multiplexing by the MV is possible because the wires to the MV behave differently at specific times (i.e., either as input or output depending on the clock phase they are in).

Figure 9.4 shows the schematic diagram of a complete QCA line-based memory cell with its Read/Write control logic. Figure 9.5 shows the QCA implementation along with the clocking zones.

The Read/Write control logic on the input side of the QCA cell consists of four gates. Two of these gates are used to determine the memory operation by ANDing the Read/Write Control signals with the Row Select signal. The other two

gates are used to duplicate the memory input signal whenever the Write Control signal is high, i.e., the MV in Zone 2 is majority dominated and its output is equal to the (new memory) input. When the Write Control signal is low, the outputs of these gates are different (zero and one respectively), such that they have no influence on the operation of the MV, i.e., the output of the MV follows the third input which corresponds to the current memory value.

The output circuitry of the memory cell consists of one gate to read the memory state depending on the value of the Control signal. As the Read Control signal must be moved to the output circuitry of the memory cell, duplication is required to allow domination of the output of the MV and the transfer of its value. A MV is required for implementing this transfer due to the clocking process which changes the direction of signal propagation. Therefore, the operation of the proposed QCA memory is determined by the following four steps:

- *Step 1:* In Step 1, the inputs and Zone 3 of the memory cell are in the Hold phase, while Zones 1 and 2 are in the Switch phase. Therefore, the QCA wires (indicated by X, Y and Z') are inputs to the MV in the Write path; Z is an output. Depending on the Write Control signal, the output Z is either the new memory input, or the old (current) memory state. For the MV in the Read path, P, Q and R' are inputs and R is an output. The output R is always equal to the inputs P and Q which correspond to the Read Control signal.
- *Step 2:* During this step, the inputs and Zone 3 of the memory cell are in the Relax phase. Zone 1 is in the Hold phase and Zone 2 is in the Switch phase. As Zone 1 is in the Hold phase, Zone 2 is in the Switch phase and Zone 3 is in the Relax phase, then the previously defined outputs Z and R now become inputs to the MV and the previously defined inputs Z' and R' become the new outputs. The input values Z and R are transferred to the outputs Z' and R' because the other two inputs of the MVs have no influence or are equal to Z and R.
- *Step 3:* During Step 3, Zone 1, Zone 2 and Zone 3 are in the Relax, Hold and Switch phases, respectively. So, a new multiplexed memory state and the Read Control signal are transferred to Zone 3.
- *Step 4:* During this step, Zone 3 is in the Hold phase and the new memory state is read at the Out cell (depending on the value of the Read Control signal).

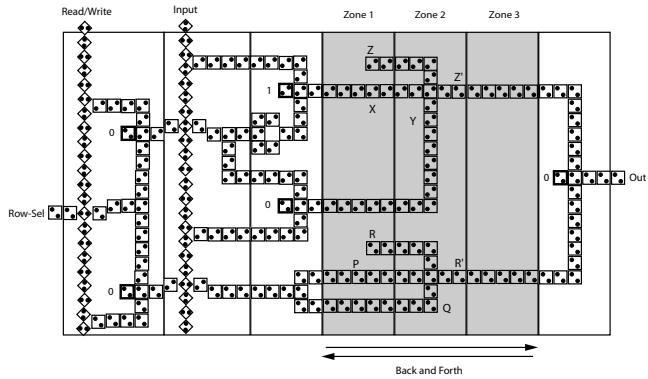


Figure 9.5 Multiplexer Circuitry to One Cell of Line-Based Memory

9.3.2 Clocking Considerations

Metal QCA designs and architectures presented in the technical literature are clocked through a single four-phased clock signal. Designs are partitioned into clocking zones; the clock signal for adjacent QCA zones is phase-shifted by $\frac{\pi}{2}$ such that a concatenation of sets of four adjacent zones is allowed as a basic mode of operation of logic propagation for the QCA circuit. However, in the proposed memory architecture two of the three zones for the memory cell are in the Switch phase at the same time. Similarly, they are in the Hold and Relax phases simultaneously. The period of the Hold and Relax phases is different for the three clocking zones. Therefore, the three zones will have to be clocked by separate signals through the underlying CMOS circuitry.

Figure 9.6 shows the periodic signals required to clock the three zones of the memory cell. In Step 1, the clocking signals of Zone 1 and Zone 2 are in the Switch phase and Zone 3 is in the Hold phase. This allows the new memory input value and the old memory state to be voted to write a new memory state. In Step 2, the new memory state in Zone 1 is transferred to Zone 2; so, the clock signals for Zone 1, Zone 2 and Zone 3 must be in the Hold, Switch and Relax phases respectively. However, Zone 2 was in the Switch phase in Step 1 and the Switch phase cannot be followed yet by another Switch phase. Therefore, Zone 1 has to be in the Hold phase long enough for the clock signal of Zone 2 to transition through the remaining phases and return to the Switch phase, at which time a new memory state is transferred. In Step 3, the memory state is transferred to Zone 3; this requires

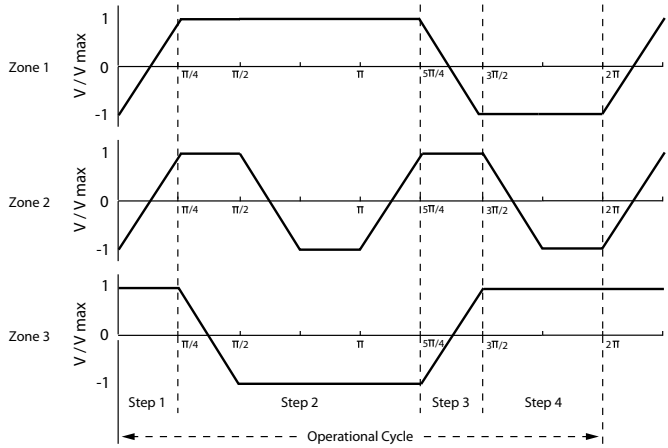


Figure 9.6 Clocking Signals for the Three Zones

all three zones to be in the Release, Hold and Switch phases respectively. In Step 4, Zone 3 is in the Hold phase such that the memory state can be read out. Zone 3 must be in the Hold phase long enough to allow Zone 1 and Zone 2 to cycle through their remaining phases and be in the Switch phase when the memory operation returns to Step 1. Zone 3 is in the Hold phase during Step 1 of the memory operation.

The cloking signals of all three zones of a memory cell are periodic with the frequency of Zone 1 and Zone 3 as half of Zone 2. All four steps of the clock signal for Zone 2 are of the same duration (i.e., the same as the global signal used to clock the remaining parts of the QCA design). Therefore, the proposed memory design requires two extra signals, to clock under the proposed parallel architecture. The input and output control circuitry is a simple combinational logic with no feedback loop and therefore, it can be clocked using conventional QCA schemes and its four equal-phased clock.

Figure 9.7 shows the CMOS circuitry required to supply the clocking signals to the proposed QCA memory design. Three periodic wave generators are used to obtain the required QCA clock signals. One of the signals is the conventional QCA clock signal (four-phased, equally timed) for the control circuitry (Read, Write), Zone 2 of the memory cells as well as the remaining parts of the QCA design. The other two clock signals which have unequal durations (as required to obtain logic propagation), are used for Zone 1 and Zone 3 of the memory cells. All columns of memory cells function identically and the clock signals from the

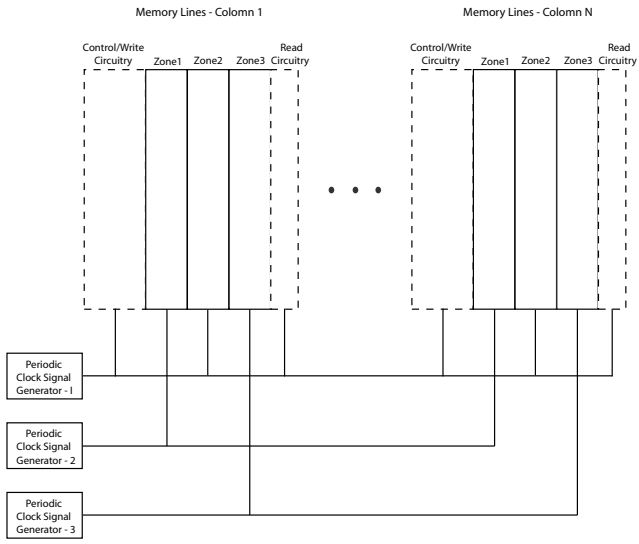


Figure 9.7 Underlying CMOS Circuitry for Clocking the Parallel QCA Memory

three wave generators are used for all of them. As can be observed from Figure 9.7 only complexity with respect to clocking is in generating the two extra signals. Routing of these signals is simplified due to regularity of the proposed design and its clocking zones.

9.3.3 Discussion and Comparison

The architecture presented in the previous Section has provided a new QCA design for parallel memory. At cell-level, the set of three clocking zones (as required for the line-based implementation of the memory cell) is applicable to all cells that are in the same column. This two-dimensional parallel architecture is similar in many respects to the 2D architecture presented by [4], although the implementation of the memory cell is radically different. Therefore this parallel memory architecture can be used for comparison purposes to evaluate the proposed memory architecture.

The memory design presented in [4] consists of a closed QCA wire loop in each memory cell, data is retained by continuously moving it in the wire loop. This mechanism requires the wire loop to be partitioned into four clocking zones, each zone is clocked by a signal $\pi/2$ shifted from the signal of the adjacent zone. As a

result, one of the four clocking zones is always in the Hold phase and data in the wire loop is retained. Apart from the clocking zones for the Read/Write logic, each memory cell requires four clocking zones. This directly translates in an increase in the underlying CMOS circuitry to provide the correct clocking signals to the zones. Also, the dimensions of these clocking zones (of width equal to that of a single QCA cell and length equal to that of few cells) make clocking extremely difficult if not infeasible. Each memory cell in this design requires a total of seven AND/OR logic gates. To reduce an MV to an AND/OR gate, the control input must be permanently set to either logic "1" or "0". The implementation of these control cells encounters an additional complexity because their polarity must be coerced to a fixed value. Moreover, each memory cell in this design requires seven such control cells.

The main advantage of the parallel architecture presented in this work is the sharing of the clocking zones between all memory cells in a column of the two-dimensional memory design. Therefore, the number of clocking zones for holding data is only dependent on the number of columns (word-size), that is, it is independent of the number of rows (memory-size). Also since clocking zones are shared, their dimensions are ideal to be clocked with underlying clocking circuitry. As the basic principle of proposed architecture is to keep memory in motion by moving data back and forth in a QCA line (rather than circulating it in a loop), a modification to the clocking process is required. In this case, the use of a one-dimensional QCA signal to clock all zones is rather restrictive. To reverse the direction of signal flow as required by the proposed architecture, clock signals with longer Relax/Hold times have to be used; this implies that two more clock signals (in addition to the conventional clock signal) are required. Table 9.2 summarizes the characteristics of the two parallel QCA memory architectures that have been compared.

As for density, Figure 9.8 shows the projected memory density for DRAM using CMOS technology and parallel memory architectures (the proposed architecture is given by the memory line curve, while the architecture of [4] is given by the Memory Loop curve). DRAM density projections are obtained from [5]. When calculating the density of a QCA memory, cell sizes of 1 and 5 nm were assumed through either molecular or metal implementation. Memory-loop architecture in [4] requires an area of $25d \times 40d$ QCA cells per one memory cell whereas the architecture proposed in this section takes an area of $25d \times 45d$ QCA cells (where d is the inter-dot distance). Overhead for additional control circuitry such as memory decoder and routing of signals is included for both architectures. Area (cost) model employed here includes any unused space within a design for calculation of total area requirements. For example, area for the memory cell in Figure 9.5 is the

Table 9.2 Comparison of Parallel QCA Memory Architectures

Characteristics	Loop-Based	Line-Based
# of QCA Cells per Memory Cell	~ 173	~ 233
# of QCA Control Cells per Memory Cell	7	5
# of Zones (Z)	4 per memory loop	3 for all memory lines in a column
CMOS Circuitry	Complex	Moderate
Clocking of Zones	Difficult	Simple

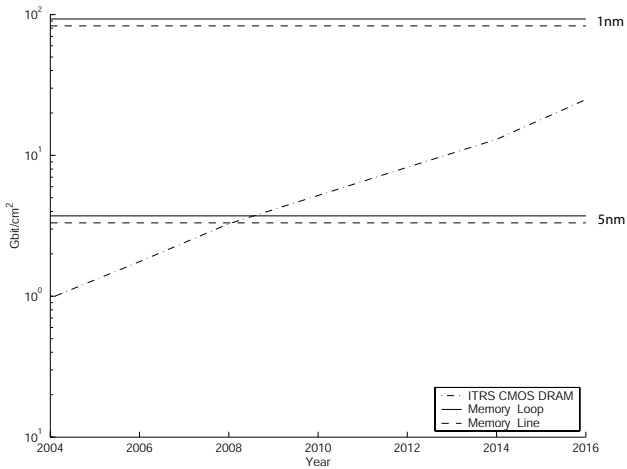


Figure 9.8 Density comparison between CMOS DRAM and parallel QCA memory architectures

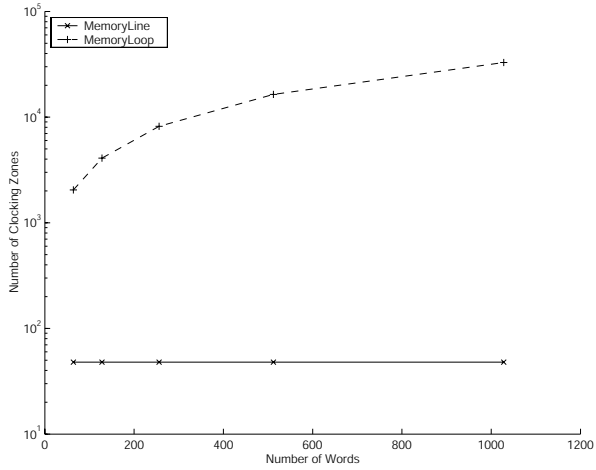


Figure 9.9 Comparison in the Number of Clocking Zones For Parallel QCA Memory Architectures

product of number of cells along X-axis times cells along Y-axis times dimension of each cell. Underlying CMOS circuitry required for clocking is considered not to create any additional overhead in this two-dimensional area calculation. From Figure 9.8, it is evident that parallel QCA memory architectures even with a metal-dot implementation (5 nm) will result in a memory density that CMOS technology will be able to match only after some years. By molecular implementation (in the 1 nm range), the large density offered by QCA for memory is further evidenced by values well beyond the range of CMOS technology. The loop-based method of [4] offers a density slightly higher than the line-based architecture of this section.

However when considering the number of clocking zones (and therefore the operating frequency of switching) the advantage of the proposed line-based architecture is substantial; for the memory-loop architecture of [4], four clocking zones are required to implement the memory loop for each bit stored in memory. For a line-based memory, only three clocking zones are required for all memory cells in a column of the two-dimensional memory array (independent of the number of rows). Therefore, in a line-based QCA memory architecture, the number of clocking zones depends only on the word width whereas in a loop-based QCA architecture it depends both on word width and the number of words. Figure 9.9 shows the number of clocking zones versus number of words for the two parallel QCA architectures

compared in this work. The difference between these two memory architectures is in orders of magnitude and increasing with the number of words.

Finally, it can be observed that the four ports (legs) of MV in the memory cell design of Figure 9.5 have different lengths. This is due to the fact that the memory designs presented in this chapter and in [4] are space optimized therefore using one dimensional clocking scheme causes uneven leg lengths for MVs. The issues related to the MVs with different leg lengths are discussed in [6]; it is stated that by including correlation terms into the solution of Schródinger equation the correct output polarization should be attained also by MVs with different leg lengths. In any case, in the proposed design, all ports (legs) of the MVs within a clocking zone can be made relatively even by simply increasing the size of the memory cell. Moreover, the issues related to uneven leg lengths could be also offset by increasing the clock period, therefore providing enough time for QCA cells to attain their true ground state [1].

9.3.4 Simulations

QCADesigner [7] provides design and simulation environment for QCA circuits. It has multiple simulation engines and standard CAD capabilities. However, this CAD tool only provides capability to clock circuits using conventional four-phased clock signal (as explained in Chapter 3) and QCA signal propagation is uni-directional. The same issues are also encountered with other available CAD tools for QCA, AQUINAS [8]. Memory cell design provided in this chapter requires multiple clock signals with different Hold, Relax times and bi-directional signal propagation.

Therefore in order to simulate the proposed memory-cell using existing CAD tools for logic verification, some design modifications are made as follows. A time-to-space transformation is performed by duplicating the circuit to make the back-and-forth movement of a memory bit over one operational cycle forth-and-forth. To simulate the memory cell over multiple operational cycles, the circuit would have to be duplicated multiple times and connected together so as to form an iterative logic array (ILA). This approach has been first proposed in [9] for VLSI testing of sequential circuits.

Figure 9.10 shows the modified design of a proposed memory element for simulation over one operational cycle. It can be observed that, instead of using a single majority voter to move data both back-and-forth, two majority voters are employed and data is moved in only one direction. Simulations have been performed using the bi-stable engine of QCADesigner with cells dimensions of $18nm$ and dot size of $5nm$. The clocking zones have also been chosen to be compatible with the

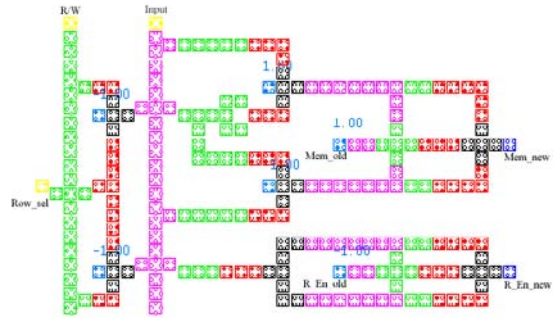


Figure 9.10 Design of Memory-Line Storage Element for Simulation Through QCA Designer

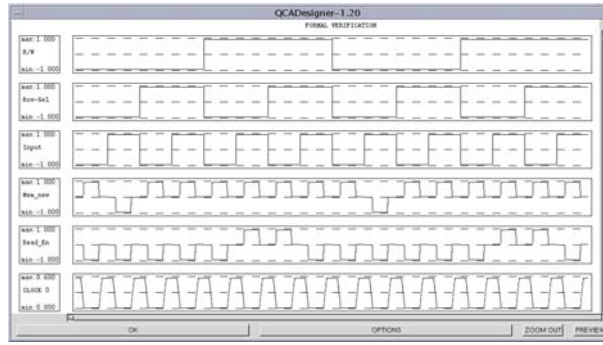


Figure 9.11 Simulation Waveforms for Memory-Line Storage Element

CAD tool. Figure 9.11 shows simulation waveforms of this modified memory cell. It can be observed that when the R/W signal and the Row-Sel are high Mem_New takes the value of Input data (after some delay) whereas at other times it is equal to Mem_Old (which is fixed at logic ”1”). Similarly Read signal is enabled when R/W signal is low and Row-Sel is high.

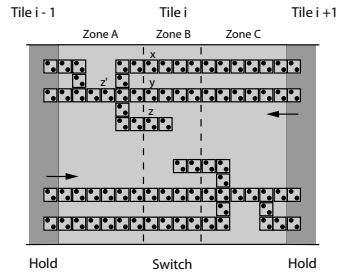


Figure 9.12 QCA Implementation of the Internal Memory Tile

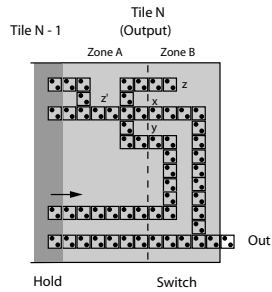


Figure 9.13 QCA Implementation of the Output Tile

9.4 SERIAL MEMORY ARCHITECTURE

9.4.1 Memory Design by Tiling

In this section, the basic principles of a novel architecture for a serial QCA memory are presented. The proposed architecture still utilizes the concept of memory-in-motion within a QCA loop. Some of the advantages of the proposed serial architecture are the novel QCA design for storing the memory bits and the associated Read/Write control circuitry. The proposed design is independent of the address decoding logic and can be used with the decoding circuits proposed for other QCA memory architectures [2] [3] [4]. QCA cells are arranged into simple basic QCA blocks referred to as *tiles*. Three types of tiles are utilized: (A) Internal memory tile (shown in Figure 9.12); (B) Output tile (shown in Figure 9.13); (C) Input tile (shown in Figure 9.14).

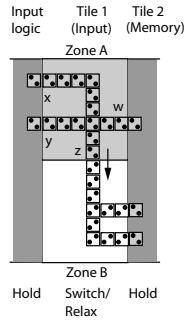


Figure 9.14 QCA Implementation of the Input Tile.

Tiles are connected in a loop using two horizontal wires (referred to as the upper and lower wires) (Figure 9.15). The memory cell in the proposed serial architecture consists of two long horizontal QCA wires connected at both of their ends by two short vertical wires, which create a loop for the memory-in-motion implementation. The Input and Output tiles and related circuits (for the Read and Write operations) are located at opposite sides of the horizontal wires. The Internal memory tiles are located between the Input and Output tiles. In this architecture, the loops are stacked, thus resulting in a highly compact memory layout. Together with a novel clocking strategy, data is allowed to move in each horizontal wire along two different directions, while still being connected into a continuous loop. Figure 9.15 shows the architecture for one memory loop using tiles; note that the size of the register is equal to the number of tiles and clocking zones are shared between all registers in the memory.

A memory loop partitioned into n tiles can store n bits of data, i.e., the number of tiles required to implement a serial memory cell is equal to its word-size. In the proposed architecture, all memory cells are arranged into a column. Tiles partition the loop of a particular memory cell and the memory loops of all other memory cells in that column. The exception is the Input tile which is used to multiplex new input values into the memory loop (and they cannot be shared with other memory cells). So, the number of tiles that are required to implement a memory of m words (for a word of b-bits wide) is given by

$$T_{m \times n} = m + (n - 1) \tag{9.1}$$

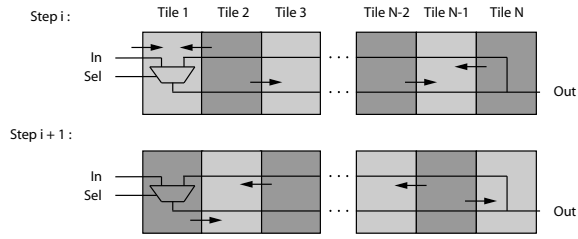


Figure 9.15 Proposed N-bit wide memory

where m corresponds to the number of Input tiles required for each of the m words and $n-1$ corresponds to the tiles that are shared between all memory cells for implementing the remaining $n-1$ bits.

For establishing the number of required QCA cells, the number of bits stored in a memory loop must be equal to the number of tiles. Therefore, the number of QCA cells required per bit is equal to the number of QCA cells of the memory loop in any tile. 74 cells are required for the Internal memory tile (Figure 9.12), whereas 24 and 54 cells are required for the Input and Output tiles (Figure 9.13 and Figure 9.14) respectively. 74 cells are required to store one bit of data, i.e., the QCA cell count is linearly related to memory size.

In the proposed serial architecture, timing and clocking are implemented using a two-level arrangement; the first level is tile-based. Each tile is divided into zones which are utilized for timing purposes for the different QCA phases. The Internal memory tile has 3 zones, the Output tile has 2 zones, and the Input tile has 2 zones. The operational cycle consists of two stages (made of multiple steps) which are tile-dependent as affecting the different zones. The second level is loop-based; each loop is partitioned into multiple columns of clocking zones; each column of clocking zones spans the same section of all loops arranged into a stack. The number of clocking zones into which the horizontal wires of the loop are partitioned determines the word size of each memory cell; the number of loops that are stacked determines the memory size.

To store data in the loops, bits move in opposite directions along the two horizontal wires. However, in the proposed architecture, similar sections of the horizontal wires are in the same clocking zone. Using a conventional four-phased clocking mechanism, data always moves in the same direction; to resolve this issue and retain the advantages of a serial architecture, tiles with different operational features must be utilized. The proposed memory is depicted in block diagram form in Figure 9.15; each tile is alternatively in two different stages (referred to as the

Hold and Switch stages) of the operational cycle, i.e., adjacent tiles are always in different stages. When a tile is in the Hold stage, it retains the bit values that are stored in the two horizontal wires of the loop and holds them as input for the next tile. When a tile is in the Switch stage, it switches to the new input bit value, thus moving data among adjacent tiles. The QCA cells in the tiles of a wire and the associated clocking strategy allow bits to move only in one direction at one time, i.e., counter-clockwise (right to left for the upper wire) for the purpose of this work. The Hold and Switch stages involve different clocking zones for the tiles and phases of the four-phased clock signal (which includes Release and Relax).

Input and Output tiles require two clocking zones per tile; all Internal memory tiles require three clocking zones per tile. Therefore, the total number of clocking zones for implementing a memory of size $m \times n$ using the proposed architecture is given by

$$Z_{m \times n} = (2 \times m) + (3 \times (n - 2)) + 2 \tag{9.2}$$

where $n - 2$ is the number of intermediate memory tiles. Therefore, the number of clocking zones required to implement the proposed architecture is very efficient. In the proposed design the maximum line length does not increase with word or memory size. Moreover, the proposed architecture retains the advantages of single-bit memory design, while clocking zones between all memory cells are shared, thus reducing the complexity of the control circuitry. A serial design has a single Read/Write logic for multiple bits in each memory cell; so, when the number of bits per cell increases, the hardware overhead per bit is also reduced.

9.4.2 Clocking and Timing

The proposed serial memory requires clocking signals that are different from the ones used in previous QCA memory designs. Signals for this architecture utilize also the same four phases for semi-adiabatic switching; but for proper clocking, the times of the signals for the Relax and Hold phases must be substantially different. All three zones in each Internal memory tile (i.e., A,B, C in Figure 9.12) as well as the other tiles do not switch in the same fashion. Therefore, multiple signals are required to clock the memory.

Consider initially the Internal memory tile. Zone 1 (A) and Zone 3 (C) of each memory tile switch identically and are always in the same phase. So, a single signal can be used to clock both of them. However, Zone 2 (B) switches differently, thus requiring a second clock signal. Although, the Output tile has only two zones (Figure 9.13), its switching mechanism is similar to the Internal memory tiles.

Therefore, its Zone 1 and Zone 2 can be clocked with the same signals used for the memory tiles. The Input tile (Figure 9.14) is partitioned horizontally and is switched differently from the memory tiles; the Input tile must multiplex the new memory state (as input) depending on the Select signal. So, the clock signal that is required to achieve this switching strategy is the same as for Zone 2 (B) of the memory tile. The Input tile requires no separate clock signal (for an Input tile, the clock signal for its Zone 2 is just a phase-shifted version of the signal for its Zone 1). Thus, two additional signals that are different from the conventional clocking arrangement of QCA are required to clock the proposed serial memory. The three signals that are required for clocking the proposed memory architecture are periodic in nature. The first half of the clock cycle corresponds to the *Switch stage*, while the second half corresponds to the *Hold stage*. Figure 9.16 shows the waveforms of the two required clock signals over one clock period (*operational cycle*).

All tiles in the memory architecture (including the Input and Output tiles) are in one of the two stages of the operational cycle, i.e., Switch and Hold. When a tile is in the Switch stage, its adjacent tiles are in the Hold stage. As all tiles alternate in stages, then during the kth (k+1) operational cycle all Internal memory tiles with even index are in the Switch (Hold) stage, while Internal memory tiles with odd index are in the Hold (Switch) stage.

The operational cycle consists of two so-called stages. For an Internal memory tile, the two stages consists of four Steps (two per stage) as follows:

- *Switch Stage:* In the Switch stage (Step 1, $0 - \frac{\pi}{5}$), all zones of the tile are in the Switch phase. At the same time, the neighboring zones are in the Hold stage ($\pi - \frac{6\pi}{5}$) and act as inputs. Hence, the input values are multiplexed and the output is moved to Zone 2 of the Switch tile. During Step 2 ($\frac{\pi}{5} - \pi$), Zone 2 is retained in the Hold phase and Zone 1 and Zone 3 are cycled through the remaining phases and returned to the Switch phase. At the same time, the neighboring zones that are in Step 4 of the Hold stage ($\frac{6\pi}{5} - 2\pi$) are released and retained in the Relax phase. Therefore, the new multiplexed values of Zone 2 are propagated to one of the desired zones (either Zone 1, or 3).
- *Hold Stage:* During Step 3, Zone 1 and Zone 3 are retained in the Hold phase such that they act as the input for adjacent tiles (which are now in the Switch stage). The old memory values (corresponding to the Switch stage in Zone 2) are released. During Step 4, all zones are returned to the Relax phase such that they can switch together again at the beginning of the next operational cycle.

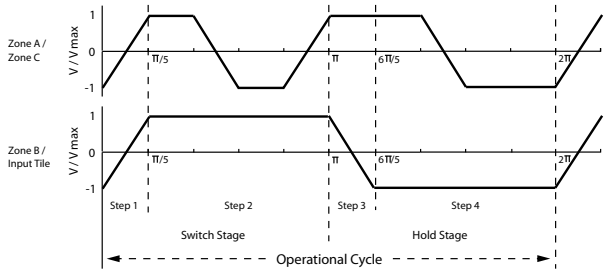


Figure 9.16 Clock Signals for Required Switching Mechanism

For an Input tile, its stages (made of 4 steps) are as follows. During Step 1 in the Switch stage, Zone 1 is in the Switch phase (the neighboring zone of the memory tile on one side and the new input signals on the other side are in the Hold phase). Since, the clock signal for Zone 2 is $\frac{3\pi}{5}$ phase-delayed of Zone 1, then Zone 1 is in the Relax phase; the MV in Zone 1 multiplexes the input value and the old memory state, thus resulting in the new memory state. During Step 2, Zone 1 is retained in the Hold phase until Zone 2 reaches the Switch phase; so, the new memory state is propagated from Zone 1 to Zone 2. At this time, the neighboring zone of the memory tile is in the Relax phase. During Step 3 (corresponding to the Hold stage), Zone 2 is in the Hold phase and the new memory state is propagated to the adjacent memory tile (which is in the Switch stage). In Step 4, both zones in the Input tile are returned to the Relax phase.

The Output file has the same operational cycle as an Internal memory tile, so the description of its operational cycle is omitted.

9.4.3 QCA Tiles

Consider initially the memory tile of the proposed QCA architecture. Each memory tile consists of a column of three clocking zones spanning the internal section of the two horizontal wires; this is applicable to all memory cells which have been arranged into a stack. When a tile is in the Switch stage, then its two adjacent tiles are in the Hold stage; therefore, each horizontal wire in the Switch stage tile has two inputs that are in the Hold phase at each of its two ends. To have a counter-clockwise memory motion, the upper horizontal wire of each memory loop must be multiplexed with the two inputs, so it transfers the input from the previous tile (on the right) to the next tile (on the left). As the lower horizontal wire is multiplexed, then it transfers the input from the tile on the left to the tile on the right. This

switching mechanism can be achieved through a MV that functionally acts as a diode (i.e., blocking the movement of data) using the clocking zones available in each tile. This MV is placed in the clocking zone near the input whose value must be dominated (masked) to obtain the unidirectional memory motion. The input whose value must be transferred is duplicated and connected to two inputs of the MV, while the input whose value must be blocked, is connected to the third input only. Therefore, the two input values are multiplexed by the MV and the output is the desired value. Using the MV, this new output is forced back to the wire whose input value was blocked. At the beginning of the Switch stage, the horizontal wires of a memory loop have the two inputs at both ends, however at the end of the Switch stage, the wire has only the value of the required input.

Figure 9.12 shows the QCA circuitry in a memory tile (referred to as the Internal tile, or tile i) for the two horizontal wires of a memory loop. This tile operates over an operational cycle made of two steps.

- *Step 1:* All three clocking zones (A, B, C) of memory tile i in the Switch stage are in the Switch phase, while clocking zones of adjacent tiles (i-1 and i+1) are in the Hold phase. Therefore, the desired input that is duplicated and connected to two inputs of the MV (i.e., x, y), is moved to the output (z').
- *Step 2:* The middle clocking zone of tile i (i.e., zone B) of the Switch stage is kept in the Hold phase, while the other two zones (A and C) are cycled through their phases and returned to the Switch phase. At the same time, the adjacent two tiles (i-1, i+1) are relaxed from their Hold phase. Therefore, the direction of signal propagation changes; a wire which was previously an output for the MV, now becomes an input and vice versa. However, two wires still remain as inputs to the MV; as the three inputs of the MV have the same signal, then the output follows this value. The output is fan-out, i.e., this signal is duplicated and propagated to the next tile during the subsequent operational cycle.

By the end of Step 2, all cells of the upper horizontal wire align to the input from the tile on the right, while blocking the input of the left tile; the lower horizontal wire aligns to the input from the left tile, while blocking the input from the right tile. Thus in one operational cycle, data in the memory loop moves by one tile in a counter-clockwise direction (right to left). In the next operational cycle, the tile which was in the Switch stage with new data on the horizontal wires, is changed to the Hold stage and the two adjacent tiles (which were in the Hold stage) are changed to the Switch stage, thus enabling further motion of data.

Two additional tiles are required for the input/output of the memory loop as well as connecting the upper and lower horizontal wires.

- *The Output tile:* the Output tile propagates data in the lower horizontal wire to the output read logic. It also has a vertical QCA wire to transfer data to the upper horizontal wire, such that the loop is established. However when the Output tile switches to accept a signal from the lower horizontal wire (which is in Hold stage), then the upper horizontal wire is also in the Hold stage. Therefore, duplication of the lower horizontal wire (to dominate the upper wire) and an appropriate switching strategy for the memory tiles is required. As the Output tile performs only one transfer (i.e., from the lower to the upper horizontal wires), so it only requires two clocking zones and one majority voter.

  Figure 9.13 shows the QCA implementation of the Output tile; the operational cycle of the Output tile is the same as that of the internal memory tile. In Step 1, the MV is switched and due to the duplication, the signal of the lower horizontal line dominates and is transferred to the output. In Step 2, the previous output line acts as an input and transfers the signal to the upper horizontal wire. Having aligned the signal on the lower horizontal wire, in the next operational cycle, the Output tile alternates to the Hold stage and loops the signal to the upper horizontal wire.

- *The Input tile:* The Input tile has a vertical wire to connect the two horizontal wires; this transfers the signal between them, such that the memory loop is constructed at the other end too. However, it is different from the Output tile, because prior to transferring data, the old memory state has to be multiplexed with the input data based on the Write Control signal for acquiring the new memory state. Multiplexing is achieved through a MV as shown previously. However, the Input tile can be affected if no horizontal partitioning is implemented for timing. The implementation of the Input tile is shown in Figure 9.14. Since the tile is horizontally partitioned into two clocking zones (upper and lower), then they cannot be shared with other memory loops. The functionality of the Input tile over one operational cycle is also given by a two-step process.

  1. *Step 1:* the lower clocking zone of the Input tile is in the Relax phase, the upper clocking zone is in Switch phase. Its two inputs (from the memory tile on one side and the Write circuitry on the other side) are in the Hold phase. Hence, the output of the MV in the top zone switches to the new memory state as input value.

2. *Step 2:* the upper clocking zone is kept in the Hold phase and the lower zone is in the Switch phase (while keeping the adjacent memory tile in the Relax phase). Therefore, the new memory value is propagated to the QCA wire connected to the lower horizontal wire. In the next operational cycle, the Input tile is in the Hold stage and the new memory value is moved back to the memory tile which is changed to the Switch stage.

As an Input tile must implement the logic to multiplex between the old memory value and the new input value, then it requires two separate clocking zones.

9.4.4 Simulation

QCADesigner [7] provides a design and simulation environment for QCA circuits; it has multiple simulation engines and CAD capabilities. This tool has been used to verify the design of the proposed QCA memory cell. A QCA memory loop that consists of an Input tile, one Internal memory tile (1 bit), and an Output tile (Figure 9.17) was assembled. As the Input and Output tiles store one bit each and the Internal memory tile stores two bits, the size of the memory cell of Figure 9.17 is 4 bits. Simulation has been performed using the bistable engine of QCADesigner with cell dimension of $18nm$ and dot size of $5nm$. However, QCADesigner does not support the clocking scheme and clock zone partitioning of the proposed memory cell; therefore, minor adjustments (mostly of a functional nature) were implemented to establish compatibility with this CAD tool. The proposed QCA memory cell has been evaluated and simulated for logic and clocking (timing) verification. In both cases, minor modifications were required; these modifications are introduced only for compatibility with QCADesigner, i.e., the modified memory circuit/clocking is isomorphic to the proposed circuit/clocking scheme, as presented in previous sections.

For *logic verification*, Figure 9.18 shows the simulation results for the memory cell; the phase of the output waveform is shifted by two clock cycles with wrap-around (i.e., the output waveform for clock cycles 7 and 8 is shown in clock cycles 1 and 2 too). During the first four clock cycles, the output is determined only by the two inputs which are connected to the two legs of the MV in the Input tile; the third leg of this MV is not connected, because it takes four cycles for the first bit to loop through the memory cell. Therefore, during this period the MV behaves as an AND gate. For the next four clock cycles (labelled five through eight in the waveform diagrams), all three legs of the MV are connected (active) and the MV inputs are

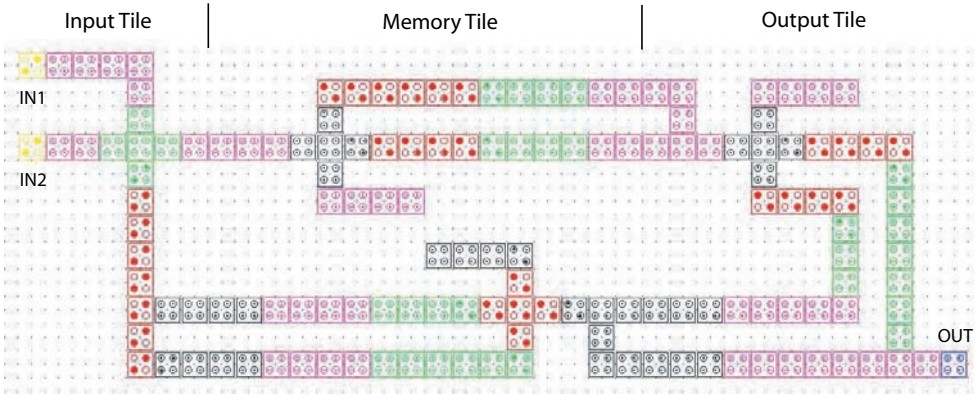


Figure 9.17 Design of Proposed Memory Cell for Simulation by QCADesigner (Logic Verification)

000, 010, 100, 111. Hence, this will result in a majority function with an output of the same value as during the first four clock cycles. As observed in the simulation results, the QCA circuit behaves as expected, i.e., data is looped in a correct manner through the tiles.

For *timing verification* of the QCA memory cell, a slight modification to the clocking strategy must be performed because QCADesigner only provides the capability of clocking circuits using a conventional four-phased clock signal as shown previously in Chapter 3. This limiting feature is also found in other CAD tools for QCA, such as AQUINAS [8]. In the proposed clocking strategy, the clocking zones next to a zone in the Switch phase must be in the Hold phase, so that all three legs of the MV can be driven. To simulate this feature, the arrangement shown in Figure 9.19 was utilized. The third leg of each MV in the Internal memory and Output tiles is permanently set to a value and placed in the same clock phase as the other two legs of the MV. The value of this third leg is dominated by the duplicated value on the other two legs, and the resulting output of the MV is propagated through the memory loop. Thus, the counter-clockwise motion of data as required by the proposed clocking strategy is still achieved within the memory loop. This modified memory cell has been simulated using QCADesigner with the above mentioned configuration; the resulting waveforms are the same as the memory cell design (shown previously in Figures 9.17 and 9.18).

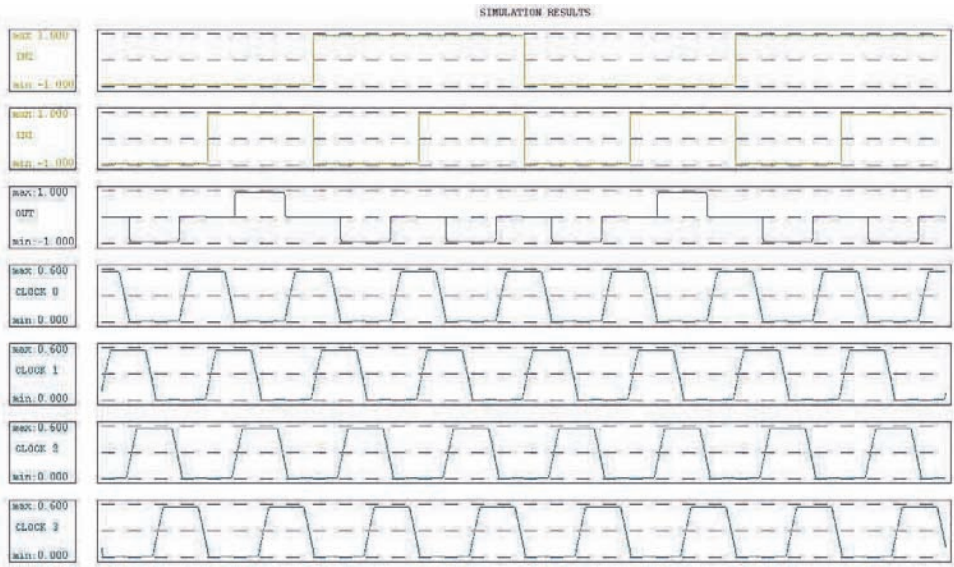


Figure 9.18 Simulation Waveforms for Proposed Memory Cell (Logic and Timing Verification)

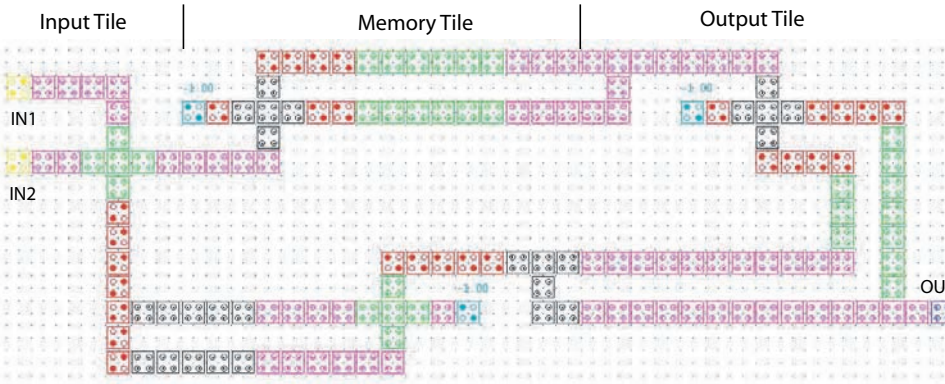


Figure 9.19 Design of Proposed Memory Cell for Simulation by QCADesigner (Timing Verification)

9.4.4.1 Comparison

In this section, an analysis is pursued to compare the proposed serial memory with other serial memories found in the technical literature [2] [3]. The serial QCA memory architecture of [3], uses a spiral (squared-shaped) that loops back to itself for storing data. The main advantage of a spiral over a loop is that sections of each layer of the spiral can be in the same clocking zone. Even though the word size of each memory cell is increased by adding extra layers, the number of clocking zones is not increased. As clocking zones span multiple layers, then their dimensions are sufficiently large to be clocked by the underlying CMOS circuitry. However, the spiral architecture of [3] has some inherent drawbacks. As the word size at each memory cell is increased by adding layers, then the number of QCA cells for implementing them increases, that is, the number of QCA cells required per data bit is not constant and depends on the word size at each memory cell.

The problem of increasing QCA cell count for additional layers leads to another drawback, i.e., the number of clocking zones into which the memory spiral is partitioned is constant. Therefore, as the dimensions of each additional layer increase, their length in some clocking zones (corners) also increases. This could be a significant problem, because the probability of kink occurrence increases with the maximum line length of a clocking zone. To avoid kinks the switching frequency must be reduced too to ensure that the QCA cells remain in the ground state.

The first significant difference between the memory spiral [3] and the tile-based memory proposed in this chapter is that the memory spiral shares clocking zones within a memory cell and hence, it is independent of word size; in the memory presented in this chapter, clocking zones are shared between different memory cells, so this scheme is independent of memory size (i.e., the number of words) but it depends on word size. As the number of memory cells is usually much larger than the number of bits in each cell, then the proposed architecture provides a better arrangement for the number of clocking zones required for timing a QCA memory. The SQUARES technique of [2] is also evaluated and compared. The modular design of this technique is different from the tiling proposed in this chapter; the basic block of [2] is designed to improve different QCA functionalities. In SQUARES, the number of clocking zones is four times the number of bits stored in memory and the number of QCA cells (per bit) needed to implement the loop is 20. The density is low because of the complex decoding and control circuitry as well as the low area utilization due to the SQUARES formalism. The tiles of the proposed approach are tailored to memory design and its performance.

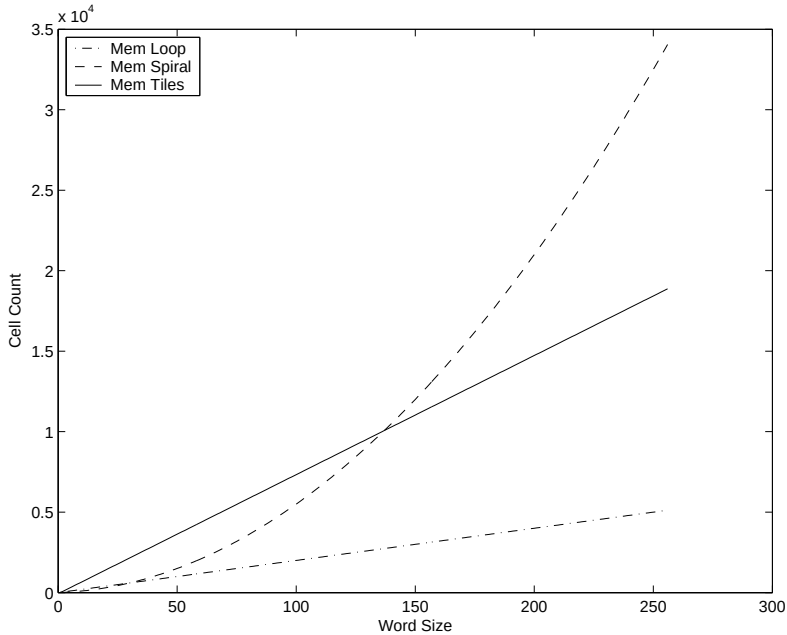


Figure 9.20 QCA Cell Count Versus Word Size

Figure 9.20 shows the cell count versus word size for the QCA memories proposed by [3] (Mem Spiral) and [2] (Mem Loop) as well as the design of this chapter (Mem Tiles). The linearity of both the proposed approach and [2] are evident even though the Memory Loop [2] requires a small number of cells. A comparison is also performed with respect to clocking zone count. Figure 9.21 shows the relationship between clocking zone count and memory size for the same three QCA architectures. In this case, [2] requires the largest number of zones, thus reducing the switching speed of the QCA memory, while the proposed scheme needs the least.

9.4.4.2 Latency Considerations

By the memory-in-motion paradigm, storage is implemented in QCA by continuously moving bits in a loop. In serial architectures, multiple bits are stored in each loop. Each memory loop is associated with a single Read/Write logic circuitry.

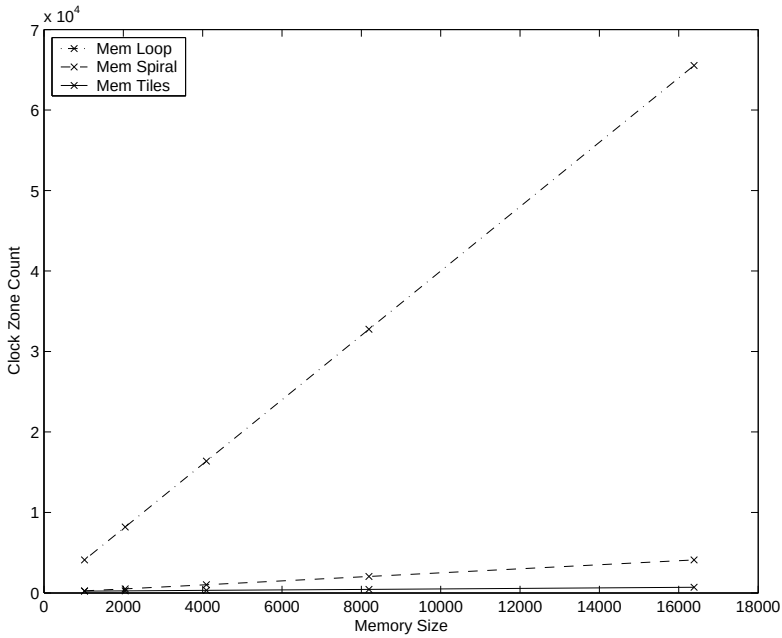


Figure 9.21 Clocking Zone Count Versus Memory Size

When the first bit of a memory word reaches this circuitry, then the memory operation can be performed, i.e., bits in the loop can be either read and transferred to an output line, or new input bits can be written into the loop. However, if the first bit passes the Read/Write circuitry, then a delay is incurred to account for cycling through the loop and returning it back to the Read/Write circuitry. On average, this delay (generally referred to as *memory latency*) is equal to half the time required to complete one revolution through the loop. However, the time required to pass through the loop depends on the loop size, which is a function of the number of stored bits (i.e., the word size). So, the word size of a memory cell must be small to reduce latency.

The serial architecture presented in this chapter is only word addressable (although, with additional circuitry also the individual bits of a word in each memory loop could be made addressable). Memory latency is incurred only for the first bit of the word, i.e., all subsequent bits are accessed in successive clock cycles with no latency. However, the serial memory designs presented in [2] are bit addressable (i.e., individual bits within the memory loop can be addressed). For random bit access, these designs incur penalty for memory latency on every bit access; therefore, if bit addressing is required, parallel QCA architectures are better suited. If serial architectures are selected for bit-addressable memories, latency considerations provide an added reason to keep the word size at each memory cell small.

For a memory design with small word size, clocking considerations (which also affect the underlying CMOS circuitry), make the serial architecture presented in this chapter more advantageous than the serial designs of [2] and [3], because clocking zones are shared between memory cells rather than within a memory cell, that is, a reduction in word size by one bit accomplishes a reduction of three in the number of clocking zones. Therefore, the reduction of word size at each memory cell and the increase of the total number of memory cells (to preserve a constant memory size) reduce the total number of clocking zones required for implementing the QCA memory. However, the QCA memory design of [3] uses a closed QCA spiral that shares clocking zones within, but not between memory cells; its reduction in word size does not reduce the number of clocking zones for the memory cell. Moreover, the reduction in word size and the increase in memory cell count result in an increase of the total number of clocking zones for the serial memory (for [2], the clocking zone requirement is rather high because it is a function of the total memory size).

In addition to the word (loop), another characteristic that affects latency, is the time for moving bits in the QCA loops. In the serial designs of [2] and [3],

one bit of the memory loop passes through the Read/Write circuitry at every clock cycle. For the proposed serial architecture, one bit passes every half cycle because the Hold stage of one tile coincides with the Switch stage of the adjacent tile (Figure 9.15). However, [2] and [3] use a conventional clock signal which has four phases in each cycle (the proposed architecture uses a different clock signal which has a repetition of four-phases for a total number of ten phases per clock cycle as shown in Figure 9.16). So, the designs of [2] and [3] require four clock-phases (one cycle) for bit movement, while the proposed architecture requires slightly more, i.e. five clock-phases (or half a cycle).

A further feature that must be considered is the time period for each phase of the clock signal. This is determined by the longest QCA line of a clocking zone [1], as

$$T_s \propto C^{1.16} \tag{9.3}$$

where T_s is the switching time for the clocking zone and C is the number of QCA cells in the longest wire of the zone. This equation shows that the dependency is nearly linear (the slightly higher exponent is attributed to approximations in the calculation).

Using the number of clock phases per bit movement and the time period of each phase, the average memory latency of the proposed scheme is given by the product of these terms times half the word size (moving the first bit to the Read/Write circuitry). Figure 9.22 shows the plot of memory latency with word size (loop size) for different serial architectures. The non linear behavior of the spiral architecture is due to the increase in word size with each additional layer and the length of the QCA line in a clocking zone. As the time period increases, then the bit movement rate is also reduced. However for the loop and tile architectures, the length of the QCA line of a clocking zone is independent of the word size and therefore, the bit movement rate remains unchanged (i.e., it is linear with word size).

9.4.4.3 Address Decoding

The circuitry that decodes signals on the address lines and selects the corresponding memory cell, is generally referred to as the *address decoder*. This is an important functional part of a memory, because it ultimately affects its performance as well as density. In the proposed architecture, the m signal lines address n memory cells, where $n = 2^m$. In traditional CMOS memories, address decoding is usually achieved through standard blocks such as m-to-n demultiplexers (such as the 74LS138), look-up tables (PROM), or programmable logic devices (PAL, PLA,

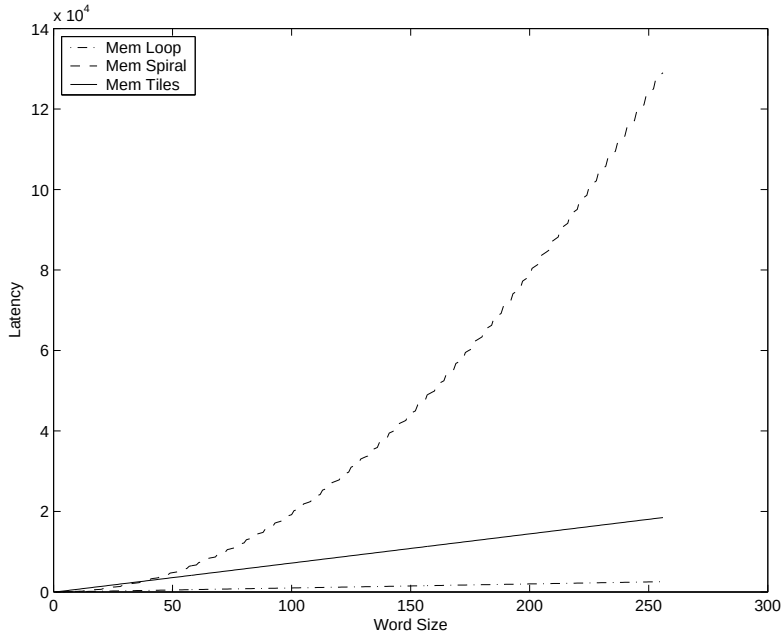


Figure 9.22 Comparison of Latency for Bit Access

FPGA). As in the early stage of research, a QCA memory will require simple circuits using combinational logic for address decoding purposes. In this section, a logic block for address decoding is presented. Issues associated with its design are discussed for improvement in reliability and performance. The characteristics and hardware requirements of the proposed architecture are then compared with other QCA decoding circuits presented in the literature.

The operation of a decoder is based on selecting one of the n Output lines by the m Select signal lines, where $n = 2^m$. Decoders usually are the preferred devices for generating mutually exclusive signals as required for addressing. In QCA, decoders can be designed using majority voters that implement the AND/OR functions. Each m-to-n decoder requires a total of $n - 1$ 2-to-1 decoders that are implemented using two MVs (as AND gates) and an inverter. Figure 9.23 shows the QCA design of a 3-to-8 decoder (in this case, $Enable$=1 and $Sel_{1,2,3}$=0, so only Out_o=1). The input A is the $Enable$ of the *memory cell* and is propagated depending on the values of the Select lines (Sel_i, i=1, 2, 3), i.e., at any given time only one of the eight memory cells which are connected to the outputs (Out_j, j=0, ... , 7) is enabled. By changing the value of A, each memory cell can be enabled/disabled.

A further issue that must be considered for address decoding, is the synchronization between accessing the memory cell and the operational cycle. The memory cycle for parallel and serial designs consists of multiple conventional (four-phased, equally timed) QCA clocking cycles. While the operational cycle of a parallel architecture consists of two QCA clocking cycles, for a serial architecture the operational cycle is made of multiple QCA clocking cycles depending on the number of bits stored in each memory cell.

The Control signals for the cells must be asserted and valid during the first clock cycle of the operational cycle of the memory when the bit (which is stored in the memory cell), reaches the input clocking zone. For a serial architecture, the Control signals must only be asserted during the first clock cycle when the start bit reaches the Input tile. If the signals are asserted in the middle of the memory cycle, the value on the input line could be written at an arbitrary position of the loop, thus corrupting the data in the memory cell.

Using address decoders, synchronization can be accomplished at relative ease by using a counter at its input. If the input of the decoder is enabled, then the Control signals to the addressed memory cell are effectively asserted; if the input is disabled, the Control signals to all memory cells (including the addressed cell) are not asserted. A counter (with a count equal to the number of QCA clocking cycles in the operational cycle of the memory) can be used to enable the decoder

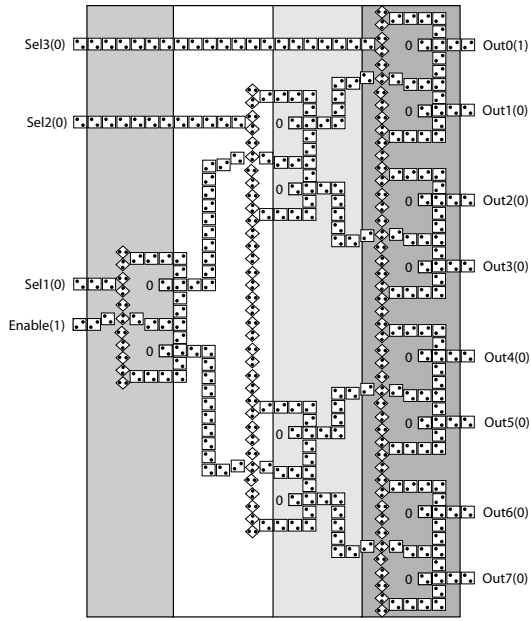


Figure 9.23 3-to-8 QCA Decoder Under One-dimensional Clocking Scheme

input at the correct time, i.e., the signals at the memory cell are asserted only at the beginning of the memory cycle. Thus, only a single counter is required to maintain synchronization for all memory cells.

The proposed circuitry can be compared with previous works. [4], for example, uses separate decoding logic for each row of the memory cells in a row-addressed two-dimensional architecture. A memory with N rows which is addressed by M (where $M = log_2 N$) address lines, would therefore require N M-to-1 decoders. As each decoder requires $M - 1$ two-input gates (for AND and OR), the total number of QCA gates (or MV) required to address the N locations using this decoding scheme is,

$$G_{M-to-1} = N \times (M - 1) = 2^M \times (M - 1) \tag{9.4}$$

The use of separate decoding to address each location (row) has an advantage in terms of latency. As it involves N M-to-1 decoders (connected in parallel), then latency in address decoding is only equal to that of a M-to-1 decoder (which requires signal propagation through $log_2 M$ levels of two input gates). So,

$$L_{M-to-1} = log_2 M \tag{9.5}$$

The decoding logic presented in this chapter uses a single M-to-N decoder which propagates the Select signal to one of the N locations (based on the signals in the M address lines). The hardware requirements for this architecture are considerably lower than for [4], which uses a separate circuit for each of the N locations. The total number of two-input QCA gates (or MV) required for the proposed decoding scheme is,

$$G_{M-to-N} = 2^{M+1} - 2 \tag{9.6}$$

However, as a single block is used for decoding the addresses of all N memory locations, then latency is also increased. As latency is related to the number of levels of two-input QCA gates (required for signal propagation to complete the decoding process by using the M-to-N decoder), then the latency is given by

$$L_{M-to-N} = M \tag{9.7}$$

Therefore, the proposed decoding circuit requires significantly less hardware for implementation, thus accomplishing a higher density (albeit it requires additional clock cycles).

For comparison, consider next the memory architecture of [3]. The H-memory structure of [3] uses a different approach for QCA signal propagation, because it

exploits micro-level pipelining in QCA wires. In previously presented designs, each bit of the memory address space is transferred through a different QCA line, similar to CMOS designs. Therefore, the decoding circuits of CMOS can be readily adapted to QCA. In the H-memory, the address and data bits are serialized and transferred through a single QCA wire, hence the decoding circuitry is substantially different. The H-memory is a complete binary tree, with memory cells at the leaf nodes and decoding logic at the root and all other internal nodes. As the address and data bits enter the structure at the root node and depending on the address value, data bits are routed to a particular memory cell; therefore, one address bit is needed for making a decision at each node.

In another serial architecture, the decoding circuitry effectively implements a binary tree with simple QCA logic gates at each node, i.e., one two-input gate in the case of the M-to-1 decoder [4] and two gates for the M-to-N decoder. In the H-memory as the address is serialized, the QCA circuitry at each node is complicated; a total of six QCA gates are required with multiple feedback loops. Therefore, the number of QCA gates (i.e., MVs) required for decoding the M bit address for the N memory cell space is given by

$$G_{M-to-N} = 6 \times (2^M - 1) \tag{9.8}$$

Latency in decoding is also increased: although the number of levels of the nodes is the same as in previous decoding designs, the complexity and computation at each node are high (in previous designs the signal must pass through a single QCA gate at each level). Therefore, latency in memory address decoding is also high.

9.4.4.4 Memory Density

Figure 9.24 shows the projected memory densities for DRAM using CMOS technology and for serial memory architectures using QCA technology. DRAM density projections are obtained from [5]. When calculating QCA memory densities, cell sizes in the range of 1 nm up to 10 nm are assumed (through either metal-dot or molecular implementations). The memory-spiral architecture of [3] requires an area of $15d \times 15d$ QCA cells per memory cell, whereas the proposed architecture takes an area of $18.5d \times 18.5d$ QCA cells (where d is the inter-dot distance). Area requirements per bit are calculated for a memory of size 256 with 12-bit words (inclusive of input/output and decoding circuitry). For decoding, the spiral memory architecture uses router cells [3], while the proposed tile-based architecture uses the decoder presented in a previous section.

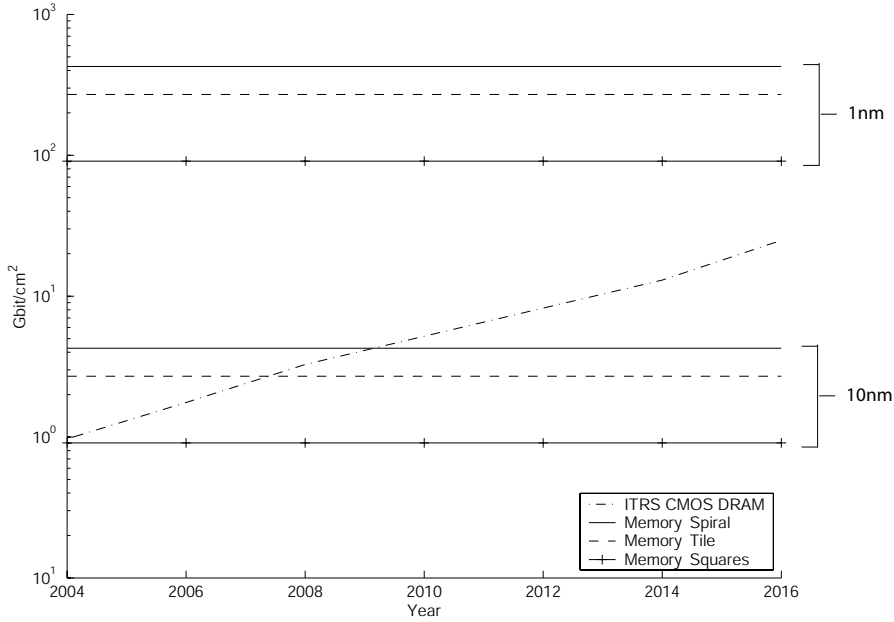


Figure 9.24 Density Comparisons of CMOS/QCA Serial Memory Architectures (Projected)

The memory architecture designed using the SQUARES formalism [4] exhibits a relatively low density. It requires an area of $32d \times 32d$ QCA cells. This low density occurs because even though the number of QCA cells for implementing the memory loop is small, there is still a substantial amount of wasted area (the goal of SQUARES is to simplify the engineering design process using uniform sized logic blocks. It has been shown that in each block the wasted area accounts for more than 50%) moreover, the feature of making data in each loop bit-addressable results in complex control and decoding circuits.

From Figure 9.24 it can be observed that the proposed serial QCA memory architecture even with metal-dot implementations (at a cell size of $10nm$) allows memory densities that can only be matched after some years by using conventional CMOS technology. For molecular implementations (at a $1nm$ range), QCA memory architectures offer incredible densities, placing them well above the range of CMOS technology.

9.4.5 Conclusion

This chapter has proposed a novel serial memory architecture for QCA implementation. This architecture is based on utilizing new building blocks (referred to as tiles) in the storage and input/output circuitry of the memory. The QCA paradigm of memory-in-motion has been accomplished using a novel arrangement in the storage loop and timing/clocking; a three-zone memory tile has been proposed by which information is moved across a concatenation of tiles by utilizing a two-level clocking mechanism. In the proposed memory, clocking zones are shared between memory cells and the length of the QCA line of a clocking zone is independent of the word size. QCA circuits for address decoding and input/output for simplification of the Read/Write operations have been discussed in detail. An extensive comparison of the proposed architecture and previous QCA serial memories has been pursued in terms of latency, timing, clocking requirements and hardware complexity. This analysis has shown that the proposed memory architecture is readily applicable to QCA implementation and provides excellent figures of merit compared with other QCA-based serial memories.

References

[1] Lent, C. S. and P. D. Tougaw “A Device Architecture for Computing With Quantum Dots,” *Proc. of the IEEE*, Vol. 85, 1997, pp. 541-557.

[2] Berzon, D. and T. J. Fountain, “A Memory Design in QCAs Using the SQUARES Formalism,” *Proceedings Ninth Great Lakes Symposium on VLSI*, 1999, pp. 166-169.

[3] Frost, S. E., et al., “Memory in Motion: A Study of Storage Structures in QCA,” *1st Workshop on Non-Silicon Computation*, 2002.

[4] Walus, K., et al., “RAM Design Using Quantum-Dot Cellular Automata,” *NanoTechnology Conference*, Vol. 2, 2003, pp. 160-163.

[5] Compano, R., L. Molenkamp and D. J. Paul, “Technology Roadmap for Nanoelectronics,” *European Commission IST programme, Future and Emerging Technologies*, also available online: *http://public.itrs.net/Files/2003ITRS/LinkedFiles/ERD/NanoeletronicsRdmp.pdf*

[6] Toth, G. and C. S. Lent, “The Role of Correlation in the Operation of Quantum-dot Cellular Automata”, *Journal of Applied Physics*, Vol. 89, 2001, pp. 7943-7953.

[7] Walus, K., et al., “QCADesigner: A CAD Tool for an Emerging Nano-Technology,” *Micronet Annual Workshop*, 2003.

[8] Blair, E. P. and C. S. Lent, “Quantum-Dot Cellular Automata: An Architecture for Molecular Computing,” *International Conference on Simulation of Semiconductor Processes and Devices*, 2003, pp. 14-18.

[9] Hennie, F. C., “Space-time Transformations”, *Finite-State Models For Logical Machines*, pp. 415-445, John Wiley & Sons, Inc, 1968.

Chapter 10

Implementing Universal Logic in QCA

V. Vankamamidi, M. Ottavi, J. Huang, M. Momenzadeh, and F. Lombardi

Design of combinational as well as sequential circuits has been explored in Chapter 4 and Chapter 8; in this chapter, the design of universal logic in QCA is presented. Initially, the design of a logic gate that implements any combinational function of at most three input variables is proposed. This type of gate is generally referred to as a universal gate and is often used as a logic resource in array structures such as FPGAs. Logic design for the universal gate with 3 inputs is initially pursued using different synthesis techniques that are tailored to QCA, namely, the AND/OR and MV-based approaches [1], as presented in Chapter 4. To extend the logic capabilities of a universal gate to an array (such as an FPGA), routing circuitry must be also considered in the design; these circuits are required for the distribution of the true/complemented signals and constant polarization signals to the universal gate. A non-blocking interconnect circuit is introduced for full connectivity of the signals.

Next, as an alternative to universal gate, the QCA designs of various look-up-table (LUT) circuits are presented. These are either memory or multiplexer-based circuits. LUTs in QCA present unique challenges because memory is implemented using the paradigm of memory-in-motion and permanent storage can be implemented using fixed polarization cells. Different memory implementations (loop and line based) are presented. Comparison between these arrangements is also pursued with respect to different figures of merit for universal gate design.

10.1 UNIVERSAL GATE

A universal gate with n inputs is formally defined [2] as a combinational circuit that can implement all possible functions with n inputs. This circuit can generate at the single output any and all 2^n product terms and the exhaustive combinations of these product terms [2] as in a SOP representation. This very powerful property is *seldom realizable*. for very large values of n due to the exponential complexity in the number of product terms; moreover, logic design in practice rarely uses combinational functions with a large number of inputs. A universal gate is also the basic logic construct by which programmable architectures can be assembled. Programmable architectures (such as FPGAs) are usually made of a homogeneous arrangement, such as a two-dimensional array and its variants [3].

To utilize a universal gate in larger designs, routing must be considered. While full connectivity of the signals is often not required, the non-blocking nature of this arrangement establishes the worst case complexity of this type of circuit. Therefore, a QCA universal gate consists of two basic types of resource:

- The *interconnect resource* consists of three fabrics: (a) a distribution network (denoted by DN); (b) K parallel 8-to-1 multiplexers (denoted by MUX); (c) a line interconnect (denoted by LI). The distribution network has 8 input lines denoted as i_h, h=1, ..., 8 corresponding to the three input signals (a, b, c) their complements (a', b' ,c') as well as the two fixed polarity values for the control cells ("0", "1"). An i_h is connected to every hth input of a 8-to-1 multiplexer, i.e., each i_h has a fanout of K. The single output line of each jth multiplexer (j=1, ..., K) is connected to the jth input line of the logic resource. The interconnect resource provides full connectivity between inputs and outputs (no blocking among signals).

- The *logic resource* may receive as inputs the outputs of the interconnection fabric (if present); the logic fabric has K input lines (denoted as I_j, j=1,2, ... ,K). K is dependent on the structure of the logic resource as generated through a logic synthesis process. Each I_j can be connected (through the interconnection fabric) to any of the i_h; so I_j can take as value any of the three literals (A, B, C), their complements (A', B', C') and two fixed value signals ("0", "1") corresponding to the fixed polarity values for the control cells in the MVs. The output is a single signal given by the output function F; F can be any combinational function of 3 variables, i.e., F=$\sum_i M_j$ where M_j is the jth product minterm (j=0, ..., 7) and $\sum_i$ denotes the ORing of the minterms (as SOP representation).

If the universal gate consists only of the logic resource, then this gate is referred to as an *unrouted universal gate* (denoted by U_U). If both resources are present, then it is referred to as the *routed universal gate* (denoted by U_R).

The process by which the logic resource of a universal gate is generated, can be thought of as an iterative procedure by which individual circuits implementing each and every specified combinational function are combined into a single circuit. A further restriction is given by the labelling of the inputs to the circuits for logic compatibility. The procedure employed in this chapter is given by the following process:

1. The circuit implementing each of the 13 standard functions f_i is generated using the selected synthesis algorithm. Let G be an empty graph, i=1.
2. Each circuit implementing f_i is described by a directed labelled graph g_i=(E_i,V_i) where the set V_i consists of MVs as vertices and the set E_i consists of the directed edges connecting the vertices as well as the primary input edges. Each primary input edge is labelled by the corresponding signal (either true literal or complemented literal or control cell binary value).
3. The isomorphism of each g_i with G is then established, i.e., whether the circuit represented by G can also implement g_i or vice versa. Modify G appropriately. Increment i.
4. If i is less than 14 go back to (2). Otherwise continue.
5. The resulting G is the unrouted universal gate.

Step 3 can be established by comparing the truth tables of the circuits and rearranging the labels of the input signals for logic compatibility. As graph isomorphism is NP complete [4], the above procedure has an exponential complexity; however for this application, the graph representation of the circuit is very simple, therefore its execution is not excessive. Moreover, for QCA graph isomorphism implies not only the compatibility of the logic input signals, but also the possible presence of constant values (0 and 1) as fixed polarity control inputs to the MV.

10.2 UNIVERSAL GATE DESIGNS

At logic-level, two synthesis approaches, namely the AND/OR-based synthesis and the MV-based synthesis [1], are applicable to QCA-based design. The details of these logic synthesis approaches have been presented in Chapter 4. In this section,

Table 10.1
Universal Gate, AND/OR-based Synthesis

	T	N	CC	CZ
DN	$251 \times 10 = 2510$	$2501 \times 9 = 22509$	0	8
MUX	$209 \times 10 = 2090$	$2090 \times 9 = 18810$	$31 \times 10 = 310$	9
LI	685	$685 \times 9 = 6165$	0	1
U_U	29	$29 \times 9 = 261$	3	3
U_R	5314	$5314 \times 9 = 47826$	313	21

the design of routed and unrouted universal gates is provided using both synthesis techniques as applicable to QCA. In all designs, the tile methodology of Chapter 7. is utilized; as the 3×3 grid is used, no additional area overhead is encountered compared with a QCA design using MV and INV gates.

10.2.1 AND/OR-based Synthesis

The unrouted universal gate as well as the implementations of all 13 standard functions using this universal gate are shown in Figure 10.1. Three clocking zones and six MVs are needed in this implementation; three of the MVs are programmed to implement the AND function by setting one of the input permanently to "0" (fixed polarization control input), so $K = 10$. This universal gate is not optimal as a three-level MV network implementation is required.

The routed universal gate is shown in Figure 10.2. Let T denote the number of tiles and N denote the number of cells ($N = 9T$). CC and CZ denote the number of control cells and clocking zones, respectively. These figures of merit are given in Table 10.1.

10.2.2 MV-based Synthesis

The universal gate generated using the MV-based synthesis approach is shown in Figure 10.3; the implementations of the 13 standard functions of [5] are also shown. In this case, U_U consists of 4 MVs (arranged in a two-level configuration), one of the MVs is always programmed to implement the AND function by setting one of the control inputs permanently to "0", so K = 8. The universal gate is shown in Figure 10.4; the hardware and timing requirements are summarized in Table 10.2.

The unrouted universal gate (generated by the MV-based synthesis and the graph isomorphism procedure presented previously) is optimal in many respects:

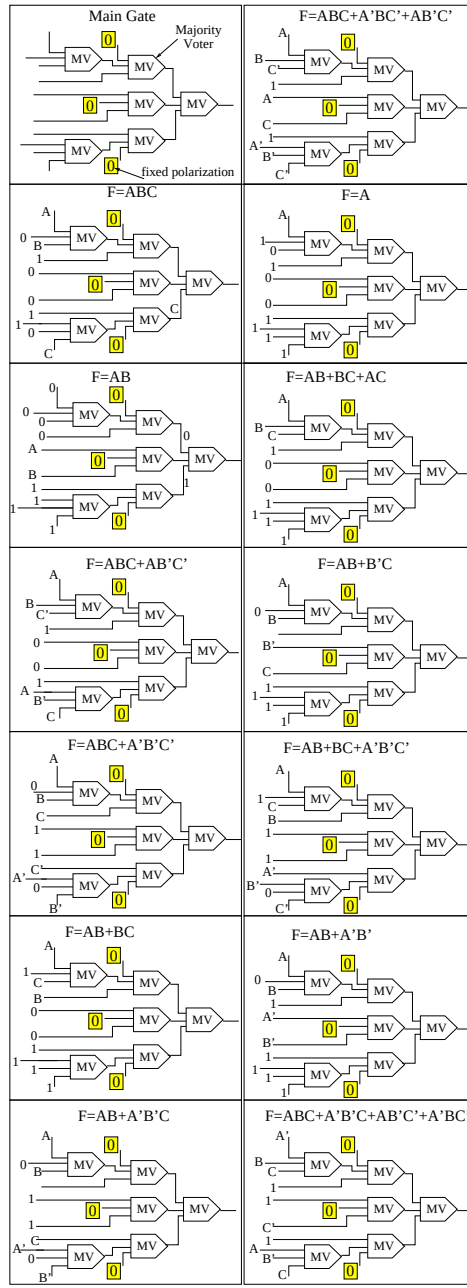


Figure 10.1 Unrouted Universal Gate and 13 Standard Function Implementations, AND/OR-based

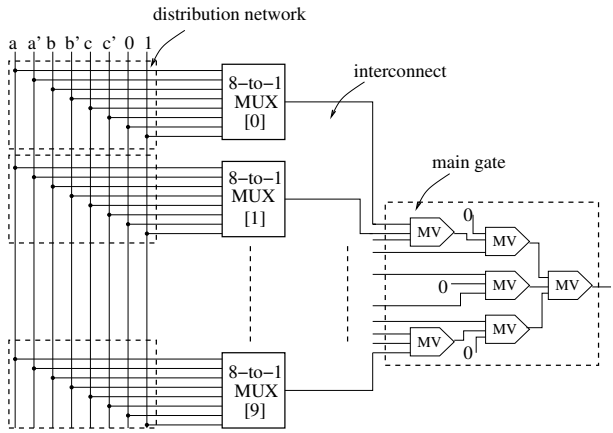


Figure 10.2 Routed Universal Gate, AND/OR-based

Table 10.2

Universal Gate, MV-based Synthesis

	T	N	CC	CZ
DN	$251 \times 8 = 2008$	$2008 \times 9 = 18072$	0	8
MUX	$210 \times 8 = 1680$	$1680 \times 9 = 15120$	$31 \times 8 = 248$	9
LI	440	$440 \times 9 = 3960$	0	1
U_U	21	$21 \times 9 = 189$	1	2
U_R	4149	$4149 \times 9 = 37341$	249	20

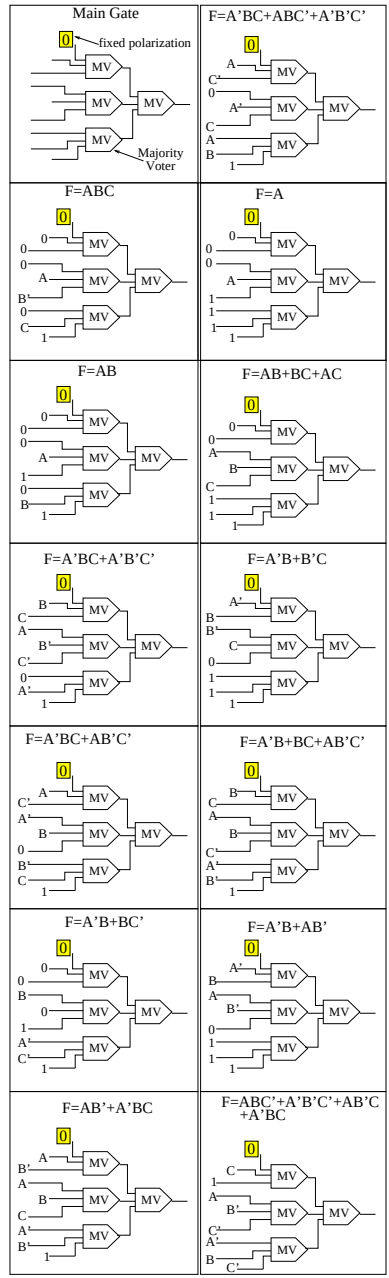


Figure 10.3 Unrouted Universal Gate and 13 Standard Function Implementations, MV-based

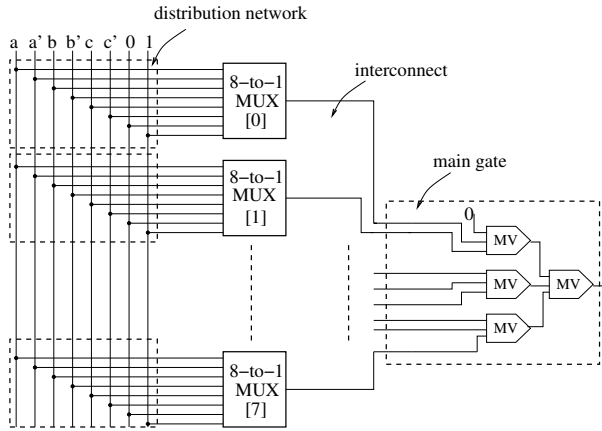


Figure 10.4 Routed Universal Gate, MV-based

1. The four MVs represent the minimum number of devices in QCA for a full two-level network implementation. Moreover, the two-level implementation also meets the criterion of minimality for combinational function for logic design.
2. The two-level network requires only two clocking/timing zones which is optimal for a two-level MV implementation.
3. The number of inputs to the unrouted universal gate is the same as the number of possible values in a QCA circuit of three input signals, that is, the three true variables, the three complemented variables and the two control values to the MVs (”0” and ”1”), so $K = 8$. This is a necessary and sufficient condition of optimality (the 3 MVs at the first level of the circuit implementation have nine inputs, of which one input is fixed to the fixed polarity value of 0).

10.3 MEMORY-BASED LUT

Other than using gates, logic universality can also be accomplished by using two types of device, i.e., memory and multiplexer. In these cases the universal logic is said to be implemented through a Look-Up-Table (LUT). A LUT offers the inherent advantage of programmability thus meeting flexibility in operation. A LUT can be either single or multiple times programmable. In the case of one-time

programmability (similar to antifused based FPGAs in VLSI), the LUT can retain its stored values when the power is turned off. For multiple programmable devices, the LUT is effectively a memory whose contents can be changed as desired by the application through the use of the two operations of Read and Write.

In this section, the design of a LUT that utilizes a parallel memory (see Chapter 9 for details on parallel memory) is proposed. This LUT consists of two blocks:

- The control and addressing logic.
- The memory array consisting of the n 1-bit storage elements.

Figure 10.5 shows the schematic of a LUT for $n = 8$ (i.e., eight 1-bit storage elements). As addressing logic, the 3-to-8 decoder selects one of the eight storage elements based on three control signals (i.e., A, B, C). The Read/Write circuitry associated with the storage elements allows these operations. The chain of OR gates connected to every storage element is used to read out data from the LUT; the storage elements that are not selected for the Read operation, generate the non-dominating value (0) to the input of the corresponding OR gate. Only the bit value of the selected storage element is provided at the input of its OR gate; this value is propagated to the end of the OR chain, i.e., at Out.

Figure 10.6 shows the implementation of a 1-bit storage element using a memory line architecture. As described previously, in this architecture data is stored using the memory-in-motion paradigm by moving the bit-value back and forth along a QCA line rather that moving it in a closed QCA wire loop. This allows sharing of clocking zones between all storage elements of the LUT (Figure 10.5) thus simplifying the underlying clocking circuitry and increasing the feasibility for implementation. The shaded region in Figure 10.6 is the storage element, while the unshaded regions are the associated Read/Write circuits.

Figure 10.7 shows the QCA design of the 3-to-8 decoder. It uses 3 ($log_2\ 8$) stages of AND gates to propagate the Select signal to one of the eight outputs based on the three control signals (A, B, C).

A similar arrangement can also be used for implementing the LUT in the memory loop architecture of [6]. The memory loop architecture [6] uses multiplexers at each storage element. In this case, the LUT shown in Figure 10.5 requires eight 3-to-1 multiplexers.

Tables 10.3, 10.4, and 10.5 summarize the hardware requirements to implement LUTs of different size (i.e., size of four, eight and sixteen 1-bit storage elements) using memory loop and line architectures. C denotes the QCA cell count. CC represents the number of control cells (the control cell is the QCA cell whose

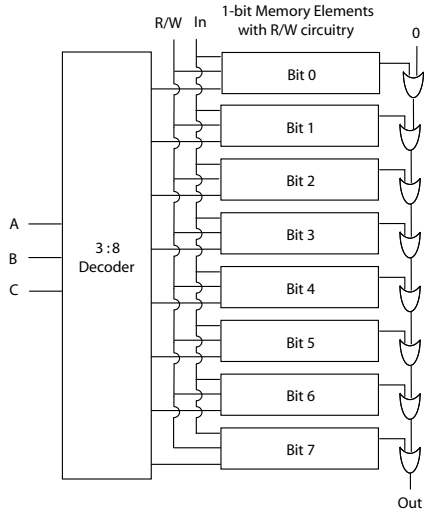


Figure 10.5 Schematic Diagram of Memory-based LUT of Size 8.

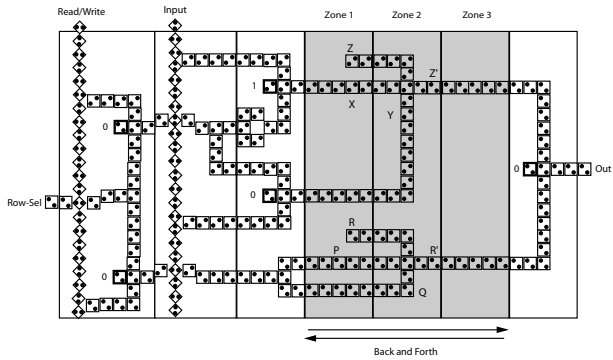


Figure 10.6 QCA Design of Storage Element for LUT Using Memory Line Arrangement

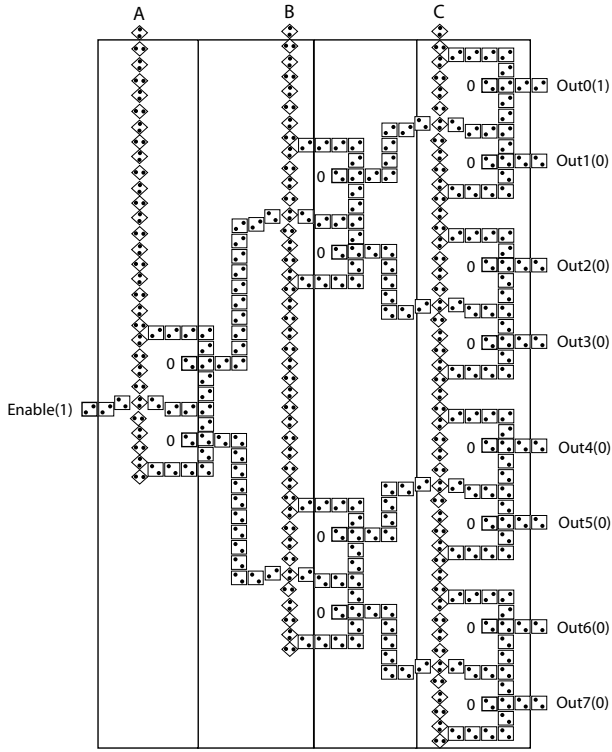


Figure 10.7 QCA Design of 3-to-8 Decoder.

Table 10.3
Hardware Requirements for 4-Bit Memory-Based LUT

	Memory line				Memory loop			
	C	CC	CZ	Area (nm^2)	C	CC	CZ	Area (nm^2)
Storage Element	932	20	7	38000	692	28	19	20800
Decoder	120	6	3	3900	152	4	2	3800
Output OR-Chain	120	4	1	5000	100	4	1	4000
Total	1172	30	11	46900	944	36	22	28600

polarity is fixed to reduce the majority voter to an AND/OR gate). CZ denotes the number of clocking zones required. The clocking zone requirements for the memory loop architecture are estimated using the clock zone partitioning scheme of [7]. Four separate clocking zones are required to implement each memory loop, thus accounting for the high clocking zone count of this architecture. The memory line architecture has a low clocking zone count, but it requires three separate clocking signals compared to the single signal for the memory loop architecture. The area (cost) model that has been employed in this analysis, includes any unused space within the Cartesian plane for calculating the total area. So for example, the area for the memory cell in Figure 10.6 is the product of the number of cells along the X-axis, the cells along the Y-axis, and the dimension and spacing of each cell. The results given in Tables 10.3, 10.4, and 10.5 are for metal-dot implementation with QCA cells dimension of $10\ nm$. If a molecular implementation with dimension in the 1 to 2 nm range is assumed, then a reduction in area by a factor of 5 to 10 nm is possible.

10.4 MULTIPLEXER-BASED LUT

A different implementation of a LUT consists of employing multiplexers. A multiplexer has n primary inputs, $log_2 n$ control inputs, and a single output. The output takes the value of the primary input whose code (address) is present on the control inputs. For a LUT implementation, the primary inputs are the 1-bit storage element

Table 10.4
Hardware Requirements for 8-Bit Memory-based LUT

	Memory-Line				Memory-Loop			
	C	CC	CZ	Area (nm^2)	C	CC	CZ	Area (nm^2)
Storage Element	1864	40	7	76000	1384	56	35	41600
Decoder	330	14	4	12200	380	16	3	19200
Output OR-Chain	240	8	1	10000	200	8	1	8000
Total	2434	62	12	98200	1964	80	39	68800

Table 10.5
Hardware Requirements for 16-Bit Memory-based LUT

	Memory-Line				Memory-Loop			
	C	CC	CZ	Area (nm^2)	C	CC	CZ	Area (nm^2)
Storage Element	3724	80	7	152000	2768	112	67	83200
Decoder	800	30	5	30700	1000	48	5	64000
Output OR-Chain	480	16	1	20000	450	16	1	16000
Total	5004	125	13	202700	4168	176	73	163200

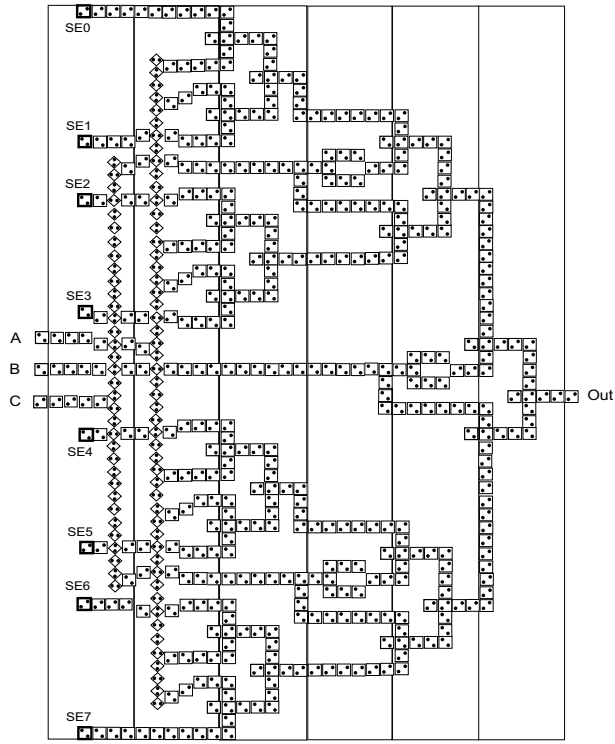


Figure 10.8 LUT of Size 8 Using Multiplexer. SE Denotes a Storage Element.

while the control inputs are the address lines for selecting the storage element. However, QCA offers an unique feature for permanent data storage, namely the use of fixed polarization cells at the primary inputs of the multiplexers. The fixed polarization cells can be programmed to take the desired logic value; however this process is not reversible.

As an example, a LUT of size 8 constructed using a multiplexer is shown in Figure 10.8; Table 10.6 summarizes the hardware requirement of multiplexer-based LUTs of different size.

Table 10.6
Hardware Requirements for Multiplexer-based LUTs

	4-Bit	8-Bit	16-Bit
C	215	575	1350
CC	9	27	61
CZ	5	6	7
Area (nm^2)	7500	20520	50400

10.5 DISCUSSION AND CONCLUSION

Using the designs presented in previous sections, the following features can be observed.

1. As an optimal technique, MV-based synthesis [1] results in an unrouted universal gate with lower hardware and timing requirements than the AND-OR-based synthesis.
2. The arrangement by which the 13 standard combinational functions can be embedded into a universal gate has been resolved using a graph isomorphic model that has resulted in a universal gate of optimal configuration (a two-level network arrangement made of four MVs and with only eight inputs corresponding to the minimum possible number of different values in QCA). By comparison, the AND/OR-based synthesis generates a universal gate of 3 levels, requiring three clocking/timing zones and 13 inputs (of which 3 are fixed polarity).
3. As established in previous sections, routing requirements for a universal gate are very cumbersome and account for a large hardware overhead. This implies that communication networks with efficient routing capabilities (probably non fully connected as in the proposed designs) must be investigated prior to a possible FPGA implementation in QCA. This result supplements and confirms the finding of [8] on the urgent need for these circuits in QCA.
4. In terms of space complexity, a memory loop-based LUT requires less area than a memory line-based LUT; however among the arrangements proposed in this chapter, a multiplexer-based LUT requires the least amount of area.

Table 10.7
Comparison for Universal Logic

Technique	Area	CZ
Mux	20520	6
AND-OR Routed	4782600	21
MV Routed	3734100	20
Memory Line	98200	12
Memory Loop	68800	39

5. CZ for a memory line based LUT is less than the CZ for a memory loop based LUT. Also with respect to this figure of merit, a multiplexer-based LUT requires the least number of CZ.
6. In a multiplexer based LUT, CZ grows as 5 + ($log_2 n$ - 2) for n = 4, 8, 16 ... The CZ in a memory line storage element and output OR chair are constant (7 and 1 respectively), but in the decoder CZ grows by 3 + ($log_2 n$ - 2).

Moreover, this chapter has shown that the QCA design of logic circuits is challenged with unique features at logic level; in this respect, it has been proved that universal gate design offers the advantage of versatility and ease of implementation using the basic QCA device primitives of MV and INV. However, due to the high complexity unrestricted and non-blocking routing to the universal gate poses severe limitations for its applicability to programmable QCA architectures such as FPGA. As an alternative, a LUT can be also utilized. For QCA, the multiplexer-based LUT design offers the advantages of high density (as requiring the least amount of area) as well as fast operating frequency (as requiring the least number of clocking zones); however it should be realized that differently from a memory-based LUT, the multiplexer based LUT is only one time programmable. These results confirm that LUTs in QCA show the same features with respect to density and operating speed as encountered in programmable VLSI circuits such as FPGAs.

References

[1] Muroga, S., *Threshold Logic and Its Applications*, New York, NY: John Wiley and Sons Inc., 1971.

[2] Biswas, N. N., "Logic Design Theory," Prentice Hall, Englewood Cliffs, 1993.

[3] Oldfield, J. V. and R. C. Dorf, *Field-Programmable Gate Arrays*, New York, NY : Wiley Interscience, 1995.

[4] Ullman, J.D., *Computational Aspects of VLSI*, Rockville, MD: Computer Science Press, 1984.

[5] Zhang, R., et al., "A Method of Majority Logic Reduction for Quantum Cellular Automata," *IEEE Trnsactions on Nanotechnology,* Vol. 3, No. 4, 2004, pp. 443-450.

[6] Walus, K., et al., "RAM Design Using Quantum-Dot Cellular Automata," *NanoTechnology Conference*,Vol. 2, 2003, pp. 160-163.

[7] Tougaw, P. D. and C. S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 75, No. 3, 1994, pp. 1818-1825.

[8] Niemier, M. T., A. F. Rodrigues and P. M. Kogge, "A Potentially Implementable FPGA for Quantum Dot Cellular Automata," *1st Workshop on Non-Silicon Computation*, Cambridge, MA, 2002.

Chapter 11

QCA Model for Computing and Energy Analysis

X. Ma, J. Huang, and F. Lombardi

One of the most pressing hurdles in the development of innovative computation paradigms and systems is energy dissipation [1]. An extensive investigation of the relation between energy dissipation and computing at logic-level has been pursued [2] with respect to the thermodynamic limit of computation. Reversible computing has been proposed to avoid this limit and improve computing power without resulting in an unacceptable energy dissipation.

QCA has been deemed as a promising technology for approaching the thermodynamic limit of computation and building reversible logic systems [1] [4]. Several QCA models have been proposed for different QCA implementation technologies [1] [5] [6]. The model in [1] has also been applied to analyze energy dissipation in QCA; [1] [4] have shown by quantitative calculation that it's possible to build reversible logic circuits using QCA.

However, the analysis of reversible logic in the context of QCA requires a substantial content of quantum dynamics. An intuitive understanding of the computation procedure and related energy dissipation is difficult to acquire due to the unique features of the quantum effects in QCA. For example, robustness to thermal effects must consider the repeated estimates of ground (and preferably near-ground) states, along with cell polarization for different designs. This evaluation is presently possible only through a full quantum-mechanical simulation. Tools such as AQUINAS [7] and the coherence vector simulation engine of QCADesigner [8] perform an iterative quantum-mechanical simulation (using the Hartree-Fock

approximation) to calculate the ground state. Other techniques such as QBert [9], Fountain-Excel simulation, and nonlinear simulation [8] only estimate the state of the cells; in some cases unfortunately, they may fail to estimate the correct ground state. These models do not fully capture the behavior of a QCA cell, as energy and related effects (such as dissipation) are not analyzed. Presently, CAD tools for QCA (such as QCADesigner and AQUINAS) are inadequate in assessing energy dissipation as related to using QCA for reversible computation. Moreover, they are applicable to an evaluation of QCA circuits under specific conditions in clocking scheme and technology implementation.

A mechanical model inspired by the operational features of molecular QCA, is proposed in this chapter. The main motivation for introducing this new model is that it provides an intuitive and classical treatment of energy and heat phenomena in QCA technology. Based on this model, reversibility of QCA is investigated in detail at both device and circuit levels. QCA devices and circuits are considered. Using this model, different features of QCA devices and circuits are analyzed. For example, the fanout connection is shown to be compatible with the reversible computing paradigm. Also, Landauer and Bennett clocking techniques [1] are briefly analyzed to unify reversibility within a cohesive framework for different QCA devices (such as the majority voter). The proposed model has the ability to evaluate different features of QCA circuits (such as clocking scheme, energy calculation, and logic state). The proposed mechanical model is currently being used as part of a CAD tool for evaluating molecular QCA; this tool is under development.

This chapter begins with a brief survey of reversible computing. The proposed mechanical model is presented next, along with a steady state analysis of QCA devices. The relationship between entropy and energy dissipation is explored in detail to present the operation of a model cell. Clocking schemes as related to reversible QCA and energy analysis are then presented. The details in deriving the mechanical model and energy anaysis is put in the appendix of the book, including a discussion on a general computing system (Appendix A), validation of the model (Appendix B), and energy analysis of small QCA circuits (Appendix C).

11.1 REVIEW ON REVERSIBLE COMPUTING

Landauer [3] has proved that the lower bound of heat dissipation is related to the loss of one bit of information during computation and is in the order of $k_B T$. Moreover, dissipation can be avoided if computation is carried out with no loss of information (this process is generally known as *reversible computing*). Intuitively, a *dynamical*

system is reversible if from any point of its state set, it is possible to *uniquely trace* a trajectory backward as well as forward in time for its computation [10]. For any thermodynamical process involving a system moving from state A into state B, the change of entropy is defined by the second law of thermodynamics as

$$S(B) - S(A) \geq \int_A^B \frac{dQ}{T}$$

where $S(A)$ and $S(B)$ are the entropy of a system in state A (initial) and B (final) respectively, and dQ is the infinitesimal amount of heat received by the system at temperature T during the change (from state A to B). The equality sign holds for a *thermodynamically reversible* process. The time reversion of a thermodynamically reversible process also satisfies the above inequality. So, to rewind a reversible process (i.e., to repeat the process from the end to the beginning in reverse order, or from B to A) does not violate the second law of thermodynamics. For a process under constant temperature, reversibility means that the total heat exchange with its environment is $T \times (S(B) - S(A))$. If this process starts and ends at the same state (i.e., a *cycle* is said to occur), then the total exchange is 0. If the cycle is not reversible, the total heat exchange is less than 0, i.e., the system is dissipative.

Landauer has shown in [3] that a computation process that loses information cannot be thermodynamically reversible. So, a computing system must dissipate heat if its working cycle consists of an information loss. To preserve information, primitives in reversible computing must have a *one-to-one onto mapping* between inputs and outputs. This property is called the *bijective property*. Primitives with this property are *logically reversible* (or invertible) primitives. The implementations of logically reversible primitives are called *reversible logic gates*, but in most cases these two words are interchangeable. Reversible computing is based on invertible primitives and composition rules that preserve invertibility [10].

The works of Bennett [11], Toffoli and Fredkin [12] [10] have shown that general computation can be accomplished effectively through a logically reversible process (i.e., without destroying or losing information). Different theoretical models of reversible computing have been proposed in the technical literature [2]. Reversibility can be analyzed in two respects:

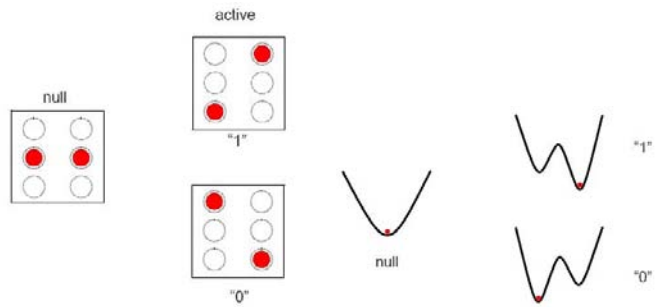
1. *Logic Reversibility:* the bijective property (one-to-one onto function) between the input and output logic states holds. This is independent of the technology and the internal structure of the circuit.

2. *Thermodynamic Reversibility:* no energy is dissipated. In this case, the internal structure of the circuit must satisfy strict reversible primitives in a given technology as an implementation platform.

Note that thermodynamic reversibility requires logic reversibility; however, a circuit can be logically reversible, but not thermodynamically reversible. In the discussion of this chapter, "reversible" means thermodynamically reversible, unless otherwise specified.

11.2 MECHANICAL MODEL

As a nano-scale device technology, the behavior of molecular QCA can be modelled using a tri-state model as proposed in [4]. The tri-state model has been presented previously in Section 3.2. In this model, a QCA cell has 2 electrons and 6 dots, as shown in Figure 11.1. This cell has three possible charge configurations: when the electrons are in the corner dots, an active state (i.e., either a 0 or a 1 state) is present in the cell; when the two electrons are in the middle dots, this represents a $NULL$ state. Inspired by this model, a novel mechanical model is proposed in this chapter for the analysis of energy and the reversible features of logic design.



(a) 3 *states* (b) *Energy states*

Figure 11.1 Tri-state Model for Clocked Molecular QCA (From [1]. ©2006 Nanotechnology. Reprint with Permission)

11.2.1 Model of QCA Cell

Figure 11.2 illustrates the computing cell in the proposed mechanical model; as shown in Appendix B, it correctly models the logic behavior of QCA devices and circuits. As shown in Figure 11.2(a), each cell consists of two units: the *rotation unit* and the *clocking unit*. A 3D view of the entire mechanical computing cell is given in Figure 11.2(b).

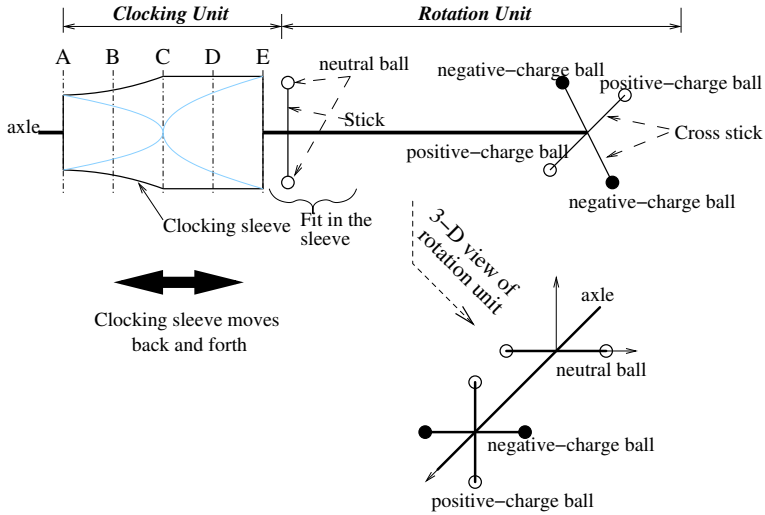
- *Rotation unit*: There are four charged balls installed at the end of a cross with four equal-length arms. Two of the balls have positive charge and the other two balls have negative charge. The charged balls are positioned to form a quadrupole, as shown in the 3D view of rotation unit in Figure11.2(a). A compressible unbendable stick connects two (neutral) balls. The center of the cross and the midpoint of the stick are installed on the same axle. The cross and the stick are tightly fixed to the axle and are always kept aligned as shown in Figure11.2(a). The position of the axle is fixed, so the only possible movement of the rotation unit is rotating around the axle.

  The angular position of the rotation unit is used to represent the information in the computing system. The charged balls in different cells interact with each other through a Coulomb force. The quadrupole interaction between mechanical computing cells can model the quadrupole interactions between cells in molecular QCA. The interaction among the mechanical computing cells is used to transfer and transform the information, in a way similar to the information processed in QCA circuits.

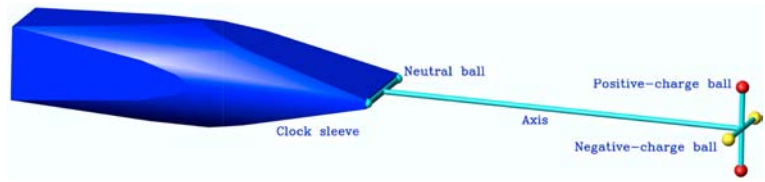
- *Clocking unit*: The neutral balls are housed in a specially shaped sleeve. The cross section of the sleeve has different shapes at different positions. Figure 11.3(a) illustrates the cross sections generated by cutting at the five positions (A)-(E) of Figure 11.2(a). The forth-and-back movement of the sleeve changes the shape that contains the neutral balls.

  The possible angular position of the rotation unit (denoted by β in Figure 11.3) is limited by the shape of the cross section of the sleeve. A large amount of energy will be required for pressing the neutral balls into the narrow part of the sleeve. This will also compress the stick connecting the balls. Thus, the position of the sleeve defines the energy state with respect to β. The plot of the energy state versus β (at the sleeve position denoted by (A)-(E)) is shown in Figure 11.3(c). The position of the charge-ball quadrupole corresponds to the degree of freedom used to encode information; this is shown under the five different scenarios in Figure 11.3(b). The sleeve interacting with

the neutral balls defines the clocking operation in the computing model, hence it is referred to as the *clocking unit*. The clocking unit works as the electrical field for QCA clocking – the electrical field also changes the energy profile of the QCA cell with a change in clocking phases. Both sleeve-clocking and QCA-clocking operate by limiting the state transitions of the cells.



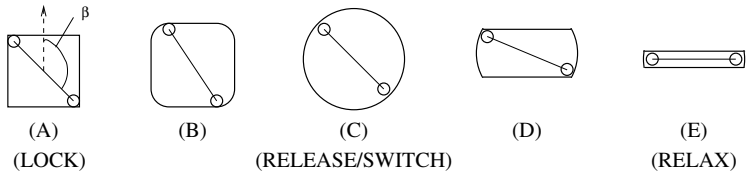
(a) Diagram of a Cell of Proposed Model



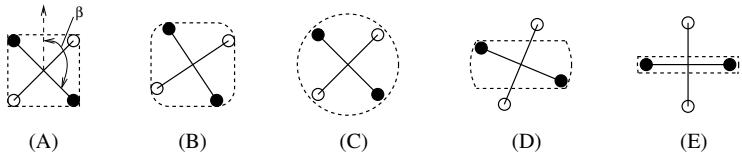
(b) 3D View of a Cell

Figure 11.2 Mechanical Model for Molecular QCA

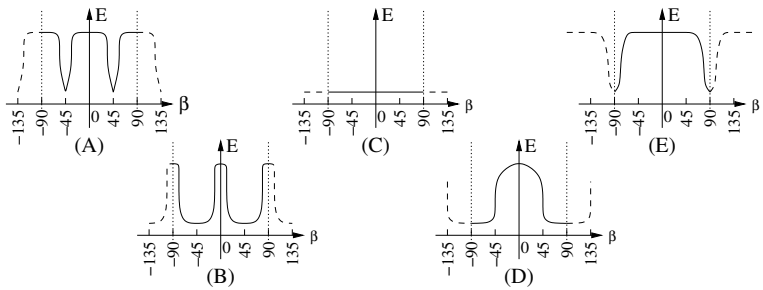
The model uses a four-phase clock configured similarly to the four-phase clock of QCA. In the LOCK and RELAX phases (corresponding to states (A) and (E) in Figure 11.3), the model precisely captures the energy state configuration of a QCA cell. In the LOCK phase, the clock sleeve constrains the angular position β of the rotation unit into two possible polarizations, 45° and 135°. Any other angular



(a) Cross−section of the clocking sleeve



(b) Position of charged balls



(c) Energy vs angular position

Figure 11.3 Clocking of the Proposed Model

position requires the stick to be compressed to fit in the sleeve. As shown in Figure 11.3(c), the energy for compressing the stick causes the energy for the position to raise rapidly when the angular position deviates from 45° or 135°. Like a QCA cell, in which electrons can only be in state "1" or "0" during the LOCK phase, the rotation unit in the proposed cell is only allowed to be in a tight range close to 45° and 135°. According to β , the two polarizations are referred to as state 1 ($\beta = 45^\circ$) and 0 (135°). In the RELAX phase, the rotation unit is allowed to be in a wide range around $\beta = 90^\circ$. This state represents the "NULL" state in a tri-state molecular QCA cell.

States (B)-(D) correspond to the SWITCH or RELEASE states in QCA. In state (C), the rotation unit is free to rotate to any angular position. It represents the state of a QCA cell in which the electrons can tunnel freely to any position. The states (B) and (D) are the transitions from (A) to (C) and (C) to (E). The rotation unit can move freely only within an angle defined by the round part shown in the sleeve's cross sections.

It is assumed that the change of the clock (as corresponding to the movement of the sleeve) is slow enough to ensure that *quasi-adiabatic switching* is applicable to the operation of the model [3]. The mechanical computing cell is filled with air, so the air behaves as a damper if the movement of the charged balls is not sufficiently slow (note that air is just a medium in the model, not a physical requirement for molecular implementation). Also, air is a source of thermal noise that gives the charged balls a random "Brownian" rotation movement.

11.2.2 Steady State Energy of QCA Devices

A QCA circuit operates by mapping the ground state to the logic solution that the circuit is designed to generate [14]. In this section, the steady state energy is calculated for several QCA devices/circuits under the proposed model. The analysis shows that the proposed model agrees with the operation of all basic QCA devices.

Assume that the size of a cell is $a \times a$; the cell center-to-center distance is denoted by b. Hereafter, it is assumed that $b = 3a$. An electric potential energy is associated with interacting charges [13]. Let two electrically charged balls in each cell be viewed as a point charge. Each positive ball has a charge $q_1 = q$, and each negative ball has a charge $q_2 = -q$. For each pair of balls at a distance of r, the potential energy is given by $E = \alpha \times q_1 q_2 / r$, where α is Coulomb's constant. To find the potential energy of a system with a set of charges, the energies associated with each pair of charges must be added. It will be shown next that for all QCA

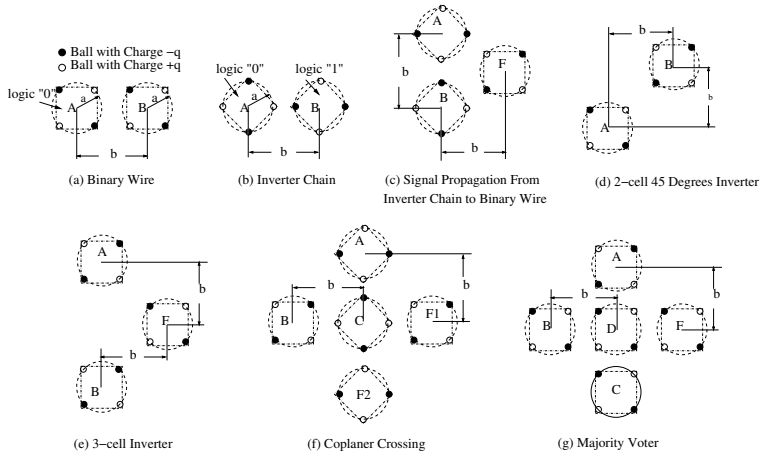


Figure 11.4 Steady State Analysis of QCA Devices

devices/circuits, this model correctly captures their operation (i.e., the lowest energy configuration corresponds to the expected function).

Binary Wire: The simplest circuit in QCA is the two-cell binary wire, as shown in Figure 11.4(a). A is the input, while B is the output. The two possible energy states, namely the aligned ($A = 0, B = 0$) and the anti-aligned states ($A = 0, B = 1$) are shown in Table 11.1. Note that by symmetry, the energy of state $A = B = 1$ is the same as the energy of state $A = B = 0$ (also the energy of state $A = 1, B = 0$ is the same as the energy of state $A = 0, B = 1$), therefore the energy of state $A = B = 0$ (also state $A = 1, B = 0$) is omitted in Table 11.1. For this device, the aligned state has the smallest energy. As expected, the two cells in the binary wire tend to have the same polarization.

Inverter Chain: By rotating the cells 45 degrees, a binary wire becomes an inverter chain, as shown in Figure 11.4(b). The possible energy states are shown in Table 11.1. It can be observed that when input $A = 0$, the lowest energy state is when output $B = 1$ (i.e., adjacent cells have opposite polarization. This is in agreement with the expected operation of an inverter chain in QCA).

Signal Propagation from an Inverter Chain to a Binary Wire: Some QCA circuits use both the inverter chain and binary wire; this is required for a circuit in which signals can be propagated from an inverter chain to a binary wire (and vice versa). The circuit that propagates a signal from an inverter chain to a binary wire is also referred to as performing a "+" to "x" conversion and is shown in Figure

Table 11.1
Steady State Energy of QCA circuits

Device	Cell State					Energy ($\times \alpha q^2/a$)
	A=	B=				
binary	0	0	—	—	—	-3.87
wire	0	1	—	—	—	-3.44
2-cell	A=	B=	—	—	—	
inverter	0	1	—	—	—	-3.99
chain	0	0	—	—	—	-3.33
+ to x	A=	B=	F=	—	—	
conversion	1	0	1	—	—	-6.093
	1	0	1	—	—	-5.536
2-cell	A=	B=	—	—	—	
45 degree	1	0	—	—	—	-3.702
inverter	1	1	—	—	—	-3.610
3-cell	A=	B=	F=	—	—	
inverter	1	1	0	—	—	-5.586
	1	1	1	—	—	-5.398
	A=	B=	C=	F1=	F2=	
	1	1	0	1	1	-9.800
	1	1	0	0	1	-9.786
	1	1	1	1	0	-9.156
coplanar	1	1	1	0	0	-9.144
crossing	1	1	1	0	1	-8.470
	1	1	0	0	0	-9.144
	1	1	0	1	0	-9.156
	1	1	1	1	1	-8.483
	A=	B=	C=	D=	O=	
majority	1	0	0	0	0	-9.57
voter	1	0	0	1	1	-9.13
	1	0	0	0	1	-9.13
	1	0	0	1	0	-8.71

11.4(c). A and B are the inputs (A and B are part of the inverter chain) and F is the output (F can then be used to drive a binary wire). Assume A=1 and B=0, then the possible energy states are shown in Table 11.1. The lowest energy state corresponds to the state in which F=1, as expected.

Inverter: In QCA, two cells placed at a 45 degree orientation will anti-align, i.e., they have opposite polarization. This structure is referred to as a 2-cell 45 degree inverter, as shown in Figure 11.4(d), where A is the input and B is the output. Let $A = 0$, then the energy of two cells placed at a 45 degree orientation is calculated using the proposed model, as shown in Table 11.1. From the calculation, the lowest energy state is when $B = 1$, in which the two cells anti-align. By symmetry, it can be shown that when input $A = 1$, the lowest energy state is obtained when

$B = 0$. Therefore in the proposed model, the 45 degree cell orientation operates as expected.

Next, consider the three-cell INV, as shown in Figure 11.4(e). A and B are the fixed inputs for $A = B$; F is the output. Let $A = B = 1$, then two possible energy states ($F = 0$ or $F = 1$) are considered. From the results shown in Table 11.1, in the lowest energy state, F has the opposite polarization of A (B). Therefore, this circuit acts as an INV.

Coplanar Crossing: The coplanar crossing circuit consists of a binary wire that crosses an inverter chain on the same planar layout. As depicted in Figure 11.4(f). A is the input of the vertical wire (the inverter chain), while B is the input of the horizontal wire (i.e., the binary wire). If A and B are fixed, the remaining three cells will have a polarization state that minimizes the total energy. Assume $A = 1$, $B = 1$, all possible energy states are shown in Table 11.1. The lowest energy state is $F1 = 1, F2 = 1$. By symmetry, the lowest energy state can be determined for the other input combinations. In all cases, the lowest energy state corresponds to the desired signal crossing state. Therefore, it can be concluded that in this model, two wires can cross each other with no interference, i.e., the coplanar crossing correctly works.

Majority Voter: If the MV (majority voter) has three identical inputs, the lowest energy state corresponds to the condition in which the device cell and the output cell have the same polarization as the inputs. The energy is calculated for the case in which the MV has inputs $A = 1$, $B = C = 0$, as shown in Figure 11.4(g). The device cell D and the output cell F will settle in a state such that the overall energy is minimized. The four possible energy states are shown in Table 11.1. The lowest energy state is the state in which the device cell $D = 0$ and the output cell $F = 0$. This corresponds to the desired MV function. The lowest energy state of other input combinations can be calculated similarly. In all cases, the lowest energy state corresponds to the state in which the device and the output cells are the majority of the inputs. Under the proposed model, the MV operates correctly.

11.3 ENTROPY AND DISSIPATION ANALYSIS

11.3.1 Operation of the Mechanical Cell

[2] has concluded that three types of a physical reversible computing model are possible: (a) Ballistic, (b) Brownian, and (c) Clocked Brownian models. *Ballistic models* need isolation between the computational system and thermal noise. In

Brownian models, the information-bearing degrees of freedom are strongly coupled to the non-information-bearing ones. In *clocked Brownian models*, the information-bearing degrees of freedom are locked and driven by the degree of freedom of a master clock, in addition to its coupling with other degrees of freedom (as in a normal Brownian model). The proposed model is a clocked Brownian model. The angular position of the rotation unit is the information-bearing degrees of freedom. It is locked and driven by the position of the sleeve. Thermal noise provides the lower bound for the energy needed to encode information (i.e., the energy involved, not the energy dissipated) in a clocked Brownian machine. When the mechanical cell is in the LOCK phase, the energy barrier separating the two possible states (also referred to as polarizations) must be bigger than $k_B T$ (where T is the operating temperature and k_B is Boltzmann's constant); this ensures that the probability of overcoming the barrier is small enough for reliable storage of information. Also, in the SWITCH phase for a cell to reliably acquire a specific state, the driver must be strong enough to constrain the rotation angle (given by β) of the rotation unit, so that its change will not exceed 90° due to thermal noise. However, the dissipation of a clocked Brownian reversible machine is proportional to the speed of computing [2] but it is not limited by such bound. If the process is slow enough to be in *quasi-equilibrium*, then the machine is capable to compute with virtually no dissipation.

Consider the entropy and dissipation in the different operations of the proposed mechanical computing cell. The analysis of the entropy is difficult if each of the three states ($NULL$, 1, 0) is defined as corresponding to a specific position of the rotation unit. In such a case, this will disable the Brownian movement of the rotation unit. A fixed rotation unit makes its changing range in angular position to zero, hence when calculating ΔS, the ratio of W_f and W_i becomes zero too. The definition of the $NULL$, 1, and 0 states must be modified so that the rotation unit can have a Brownian movement within a small angle interval given by $[-\delta, \delta]$ (Figure 11.5). Let S_0 denote the entropy of a cell in the $NULL$ state.

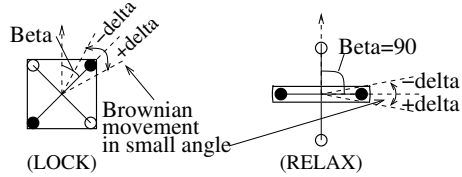


Figure 11.5 Rotation Unit With Brownian Movement at a Small Angle

The following analysis assumes a clocking unit (sleeve movement), that can store energy (a large, but still finite amount) and exchange energy with the mechanical cell with no loss of energy (this feature is not related to computing, hence it does not affect the analysis).

- First, in the mechanical model the cell is moved from the $NULL$ state to either the 1, or 0 state reversibly. A driver and the movement of the sleeve can achieve this process. With no loss of generality, we consider the 1 state in this example. Initially, the shape of the cross section of the sleeve is changed to a circle. During this change, $W = -k_B T \log_2 \frac{\gamma}{2\delta}$ of work is done to the rotation unit and the heat exchange is $Q = k_B T \log_2 \frac{\gamma}{2\delta}$ (heat flows from the environment to the rotation unit), where γ is the range of the possible positions of the rotation unit. The driver must be strong enough to limit the rotation unit in $[0, 90]$ with a high probability, i.e., $\gamma < 90$. Subsequently, the shape of the cross section of the sleeve is changed to a square. During this process, $W = k_B T \log_2 \frac{\gamma}{2\delta}$ of work is exerted to the rotation unit and $Q = -k_B T \log_2 \frac{\gamma}{2\delta}$ (heat flows from the rotation unit into the environment).

  This process is logically reversible: given the same initial state (i.e., $NULL$), then the cell goes into a final state specified by the polarization of the driver. Also, this procedure entails no dissipation. The total heat exchange between the system and the environment is zero. Work is done by the driver, the clocking unit, and the rotation unit, but the total work is zero. Although the rotation unit performs zero work in total, some energy (given by E_{p1}) [1] is transferred into the clocking unit; E_{p1} is the difference between the potential energy of the driver and a $NULL$ state cell and the potential energy of the driver and a cell in a polarized $(1/0)$ state.

- Next, if the polarization of a cell in the LOCK phase is known, then it is possible to place the cell to the $NULL$ state (when the clock goes to the RELAX phase) with no dissipation. This process needs an external driver with the same polarization as the cell. Initially, when the shape of the cross section of the (clocking unit) sleeve becomes circular, the external driver keeps the rotation unit in a range smaller than $[0, 90]$. During this step, $W = -k_B T \log_2 \frac{\gamma'}{2\delta}$ is exerted to the rotation unit and $Q = k_B T \log_2 \frac{\gamma'}{2\delta}$ (also, γ' is the range of possible positions of the rotation unit). The driver must keep $\gamma' < 90$. Subsequently, when the clock is in the RELAX phase, the state of the cell changes into $NULL$. During this process, $W = k_B T \log_2 \frac{\gamma'}{2\delta}$ and

1 This energy initially comes by applying the driver

$Q = -k_B T \log_2 \frac{\gamma'}{2\delta}$. This process is also reversible and no heat is dissipated. There is an energy (given by E_{p2}) that is transferred from the clocking unit to the driver. E_{p2} is the difference between the potential energy of this driver and a $NULL$ cell and the potential energy of this driver and a polarized cell.

Consider an entire clock period (from the RELAX phase to the LOCK phase and then back to the RELAX phase), so no energy dissipation will occur. From the RELAX phase to the SWITCH phase, the potential energy E_{p1} between the driver and the charged balls flows into the clocking unit. Then from the RELEASE to the RELAX phase, E_{p2} will flow back from the clocking unit and it becomes potential energy. E_{p1} and E_{p2} are established by the strength of the drivers during these two phases; thus, it is possible to find a *Carnot cycle* provided the strength of the driver is kept constant during these two phases, i.e. $E_{p1} = E_{p2}$. If the strength of the driver is not constant, then $E_{p1} - E_{p2}$ will flow into the clocking unit.

The reversibility of this process depends on the polarization of the external driver during the RELEASE phase. The following scenarios can be distinguished:

1. If there is no driver during the RELEASE phase, then a free expansion will occur when the cross section of the sleeve changes to a circular shape: the rotation unit increases its range of possible angular positions from $[0, 90]$ to $[0, 180]$. As in the ideal gas example presented previously, there is no work and exchange in free expansion. Prior to free expansion, $W = -k_B T \log_2 \frac{90}{2\delta}$ is done to the rotation unit and $Q = k_B T \log_2 \frac{90}{2\delta}$ (from the environment to the rotation unit). After the free expansion, $W = k_B T \log_2 \frac{180}{2\delta}$ and $Q = -k_B T \log_2 \frac{180}{2\delta}$. So, in the whole RELEASE phase, the clocking unit exerts $\sum W = k_B T$ of work to the rotation unit and the system dissipates $k_B T$ ($\sum Q = -k_B T$).

2. If the driver's polarization is different from the cell, then energy dissipation will occur. As illustrated in Figure 11.6, the driver will force the balls to the other polarization state. In the LOCK phase, the polarization change cannot occur because there is not enough energy to compress the stick to turn the rotation unit to a new polarization. However, during the SWITCH phase, when the cross section of the sleeve is changing from a square to a circle, the energy required for the polarization change is small. At this point, the rotation unit will change into the new polarization; it will receive a kinetic energy E_k from this change and vibrate around the new polarization position until damping due to the air will slow it gradually to the average thermal noise level. Damping will cause dissipation of E_k. Prior to the polarization change, due to the driver, the angle β of the rotation unit is $[90 - \gamma_1, 90)$,

where $0 < \gamma_1 < 90$. During the RELEASE phase prior to the polarization change, $W = -k_B T \log_2 \frac{\gamma_1}{2\delta}$ and $Q = k_B T \log_2 \frac{\gamma_1}{2\delta}$. Then, a free expansion process increases the possible range of β to $\gamma' = min(90 + 2\gamma_1, 180)$. In free expansion, $W = 0$ and $Q = 0$. During damping, E_k is dissipated into the environment to slow down the rotation unit. Also, the driver finally limits the range of β at an angle γ_2. During this process, $W = k_B T \log_2 \frac{\gamma'}{\gamma_2}$ and the rotation unit receives $Q = -k_B T \log_2 \frac{\gamma'}{\gamma_2}$ from the environment. After damping utill the end of the RELEASE phase, $W = k_B T \log_2 \frac{\gamma_2}{2\delta}$ and $Q = -k_B T \log_2 \frac{\gamma_2}{2\delta}$.

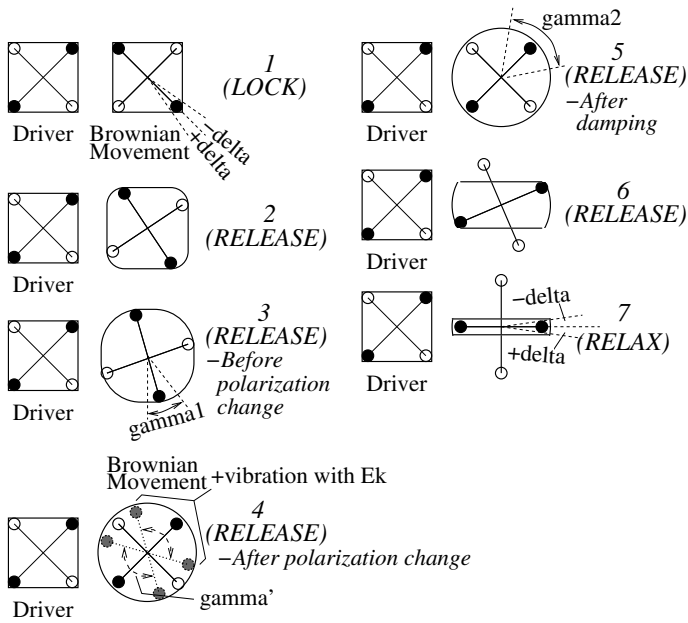


Figure 11.6 RELEASE Phase for a Cell Under a Driver of Different Polarization

So, in the entire RELEASE phase, the total work is $\sum W = k_B T \log_2 \frac{\gamma'}{\gamma_1}$ and $E_k + k_B T \log_2 \frac{\gamma'}{\gamma_1}$ is dissipated (equal to $-\sum Q$). E_k, γ' and γ_1 are determined by the strength of the driver. As the driver is strong enough to set a cell to a polarization with high probability, then it must be in the order of $k_B T$ (according to Boltzmann's distribution). Approximately, E_k is given by

$E_k = k_B T$ as $\gamma' = min(90 + 2\gamma_1, 180)$ and $0 < \gamma_1 < 90$, $k_B T \log_2 \frac{\gamma'}{\gamma_1} \geq 2$. So, the RELEASE phase dissipates at least $2k_B T$.

3. If the polarization of the cell is not known in the LOCK phase, it is impossible to utilize any reversible process to set it to the $NULL$ state. If a constant driver is applied, then there is a 50% probability to be in the same polarization as the cell, thus no energy is dissipated. However, there is also a 50% probability to be in the opposite polarization as the cell, thus at least $2k_B T$ will be dissipated. The expected dissipation is $k_B T$. If no external driver is applied, the process still dissipates $Q = k_B T$.

As suggested in Appendix A, the "free expansion" process when resetting a computing cell with no knowledge of its state is the source of the dissipation lower bound for information loss. In the analysis above, it is shown that releasing a QCA cell without a driver of equal polarization will result in "free expansion" and, in turn, dissipation.

The validation and application of this QCA dissipation analysis is presented in Appendixes B and C.

11.4 LANDAUER AND BENNETT CLOCKING SCHEMES

The clocking scheme that was assumed in the previous sections, is generally referred to as *Landauer clocking*. Landauer clocking is the scheme utilized in almost all previous QCA papers found in the technical literature. Landauer clocking is simple, however, it makes few circuits (such as the MV) to be irreversible and dissipative. In [1] [4], a different scheme (i.e., the so-called *Bennett clocking*) has been proposed for QCA, under which MV can be non-dissipative. Figure 11.7 illustrates these two types of clocking scheme. The proposed model can be used to understand the operations of the two clocking schemes. An analysis and comparison of the two clocking schemes are presented in this section.

The basic principle of Bennett clocking is that the bit information is held in place by the clock until an operation is completed by the circuit [1]. Then, it is erased in the reverse order of computation, as illustrated in Figure 11.7(b). Thus, every cell is switched and released when all other cells in the circuit are in the same configuration. It is evident that every cell has a driver of same-polarization when it is released. As per the conclusions drawn in Section 11.3.1, every cell works reversibly. So, the computing process of the whole circuit is reversible. Quantum-dynamic calculation has shown that energy dissipation per switching event is much

less than $k_B T ln2$ for QCA circuits containing the MV and fan-out [1]. Bennett clocking does not require any change in QCA layout, because only the clocking signals are modified.

However, the control of Bennett clocking is more complex compared with Landauer clocking. Additionally, in Bennett clocking the next operation cannot begin until the circuit is released from the output to the input. For QCA, the speed of a Bennett clocked computation is proportional to the timing depth (number of clocking zones) of the circuit. By comparison, Landauer clocking releases a cell after four phases (1 clock cycle) of quasi-adiabatic switching, so that the cell can be used in the next operation. Landauer clocking leads to a pipeline implementation (Figure 11.7) and an increase in computing speed. Bennett clocking releases the cells from output to input, so the last cell is locked; as for the input cells, they are released under no driver. As analyzed in previous sections, the only energy dissipation in Bennett clocking occurs when the input cells are released under no driver.

A two-to-one multiplexer (MUX) is used as an example to illustrate the advantages and disadvantages of Landauer and Bennett clocking schemes. The schematic diagram and the corresponding layout of the MUX are shown in Figure 11.8; when $Sel = 1$, $F = A$; when $Sel = 0$, $F = B$. The clocking zone assignments are the same for Landauer and Bennett clocking schemes and are represented by different shaded colors and patterns in the layout. The timing diagrams for Landauer and Bennett clocking schemes are depicted in Figure 11.9.

- If Landauer clocking is used, the delay between the inputs and the outputs is 10; consecutive inputs can be applied at every clock cycle (four clocking zones). So, consecutive outputs are produced every clock cycle.
- With Bennett clocking, the delay between inputs and outputs is again 10 clocking zones. However, consecutive inputs can be applied with a delay of 22 clocking zones, which is four times more than for Landauer clocking.

Bennett clocking results in a longer delay compared with Landauer clocking; however, the energy dissipation of Bennett clocking occurs only at the input/output ports and the internal energy dissipation of Bennett clocking can be made arbitrarily small. The energy dissipation of Landauer clocking is proportional to the number of irreversible gates in the circuit. Therefore, Bennett clocking is more energy efficient than Landauer clocking. Clearly, there is a trade off between power (and reversible computing) and delay (for high performance computing) when choosing the desired clocking scheme for a QCA implementation.

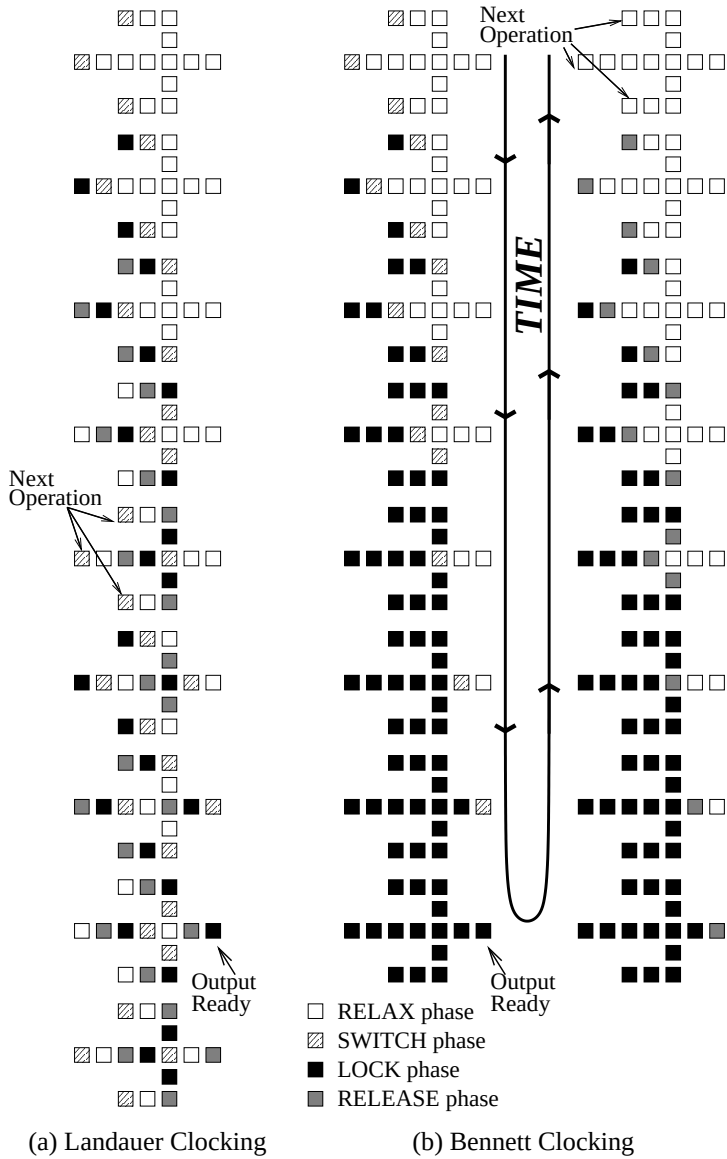


Figure 11.7 Landauer and Bennett Clocking Schemes

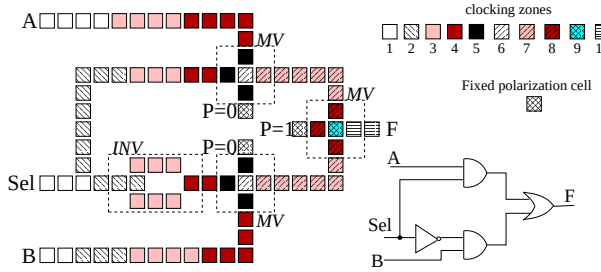


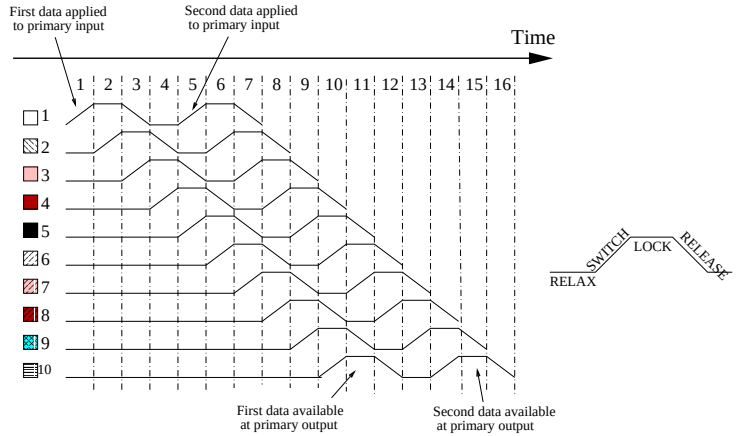
Figure 11.8 Two-to-one MUX Schematic and Layout Diagrams

11.5 CONCLUSION

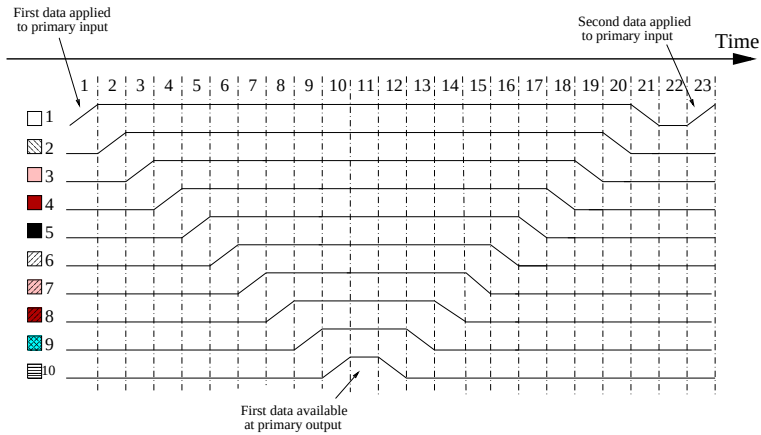
This Chapter has presented a new mechanical-based model that is amenable to QCA operation and computation. This mechanical model is inspired by the operational features of clocked molecular QCA to provide an intuitive and classical view of the energy and heat phenomena. The proposed mechanical model consists of a sleeve of changing shape; four electrically charged balls (two with negative charge and two with positive charge) are used to model the electrically neutral QCA molecule. The balls are connected by a stick that rotates around an axle in the sleeve. The sleeve acts as a clocking unit, while the angular position of the stick within the changing shape of the sleeve, identifies the phase for quasi-adiabatic switching. Recently, QCA has been advocated as a potential candidate technology for implementing reversible computing, so the proposed model can be utilized to assess these features. By avoiding a full quantum-thermodynamical calculation, it has been shown that the proposed model is versatile in evaluating different features (such as energy consumption for reversible computing and clocking schemes) at device and circuit levels for molecular QCA implementation.

The steady-state energies of various QCA devices have been calculated using the proposed model. It has been shown that the mechanical model agrees with the operation of all basic QCA devices. These results have been also confirmed by QCADesigner. The proposed model has been used to characterize the dynamic behavior of QCA circuits. It has been shown that this model is very effective in analyzing different QCA circuits for reversible computing.

- The QCA shift register (irrespective of the number of cells per stage) is a reversible circuit.



(a) Timing of Landauer clocking scheme



(b) Timing of Bennett clocking scheme

Figure 11.9 Timing Diagrams for the MUX Under Landauer and Bennett Clocking Schemes

- In contrast with other technologies, the fanout circuit in QCA does not necessarily result in energy dissipation; the increase in dissipation is the result of having an additional output cell connected to this circuit. Therefore, dissipation is associated with the irreversible release at the output cells and the reversibility of adjacent circuits.
- The 3-cell inverter is a reversible circuit.
- The majority voter circuit in QCA shows an energy dissipation dependency on the clocking scheme; MV is irreversible if Landauer clocking is used, but reversible under Bennett clocking. This confirms previous results found in the technical literature [1].
- Through the example of a two-to-one multiplexer, it has been confirmed that there is a tradeoff between energy consumption (and therefore reversible computing) and the number of clocking zones (delay) when selecting a clocking scheme for QCA.

References

[1] Lent, C. S., M. Liu, and Y. Lu, "Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling," *Nanotechnology*,Vol. 17, No. 16, 2006, pp. 4240-4251.

[2] Bennett, C. H., "Notes on the History of Reversible Computation", *IBM Journal of Research and Development*, vol 44, No. 1, 2000, pp. 525-532.

[3] Landauer, R., "Irreversibility and Heat Generation in the Computing Process", *IBM Journal of Research and Development*, vol 5, 1961, pp. 183-191.

[4] Timler, J. and C. S. Lent, "Maxwell's Demon and Quantum-dot Cellular Automata," *Journal of Applied Physics*, vol 94, no 2, 2003, pp. 1050-1060.

[5] Tang, R., F. Zhang, and Y.B. Kim, "QCA-Based Nano Circuits Design", *IEEE International Symposium on Circuits and Systems*, 2005, pp. 2527-2530.

[6] Tougaw, P. D. and C. S. Lent, "Logical Devices Implemented Using Quantum Cellular Automata," *Journal of Applied Physics*,Vol. 75, No. 3, 1994, pp. 1818-1825.

[7] Blair E. P., "Tools for the design and simulation of clocked molecular quantum-dot cellular automata circuits," Master's thesis, University of Notre Dame, Department of Electrical Engineering, 2003.

[8] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003, also available online: *http://www.qcadesigner.ca/papers/micronet2003.pdf*

[9] Niemier, M. T., M. J. Kontz, and P. M. Kogge, "A Design of and Design Tools for a Novel Quantum-dot Based Microprocessor," *Proceedings Design Automation Conference*, 2000, pp. 227-232.

[10] Toffoli, T., "Reversible Computing", *Technical Report MITLCSTM151, MIT Laboratory for Computer Science*, 1980.

[11] Bennett, C. H. “Logic Reversibilty of Computation”, *IBM Journal of Research and Development*, vol 17, 1973, pp. 525-532.

[12] Fredkin, E. and T. Toffoli, “Conservative Logic”, *International Journal of Theoretical Physics*, vol 21, 1982, pp 219-253.

[13] Fermi, E., *Thermodynamics*, New York, NY: Dover Publications, Inc., 1956

[14] Lent, C. S. and P. D. Tougaw, “A Device Architecture for Computing with Quantum Dots,” *Proc. of the IEEE*, Vol. 85, 1997, pp. 541-557.

Chapter 12

Fault Tolerance of Reversible QCA Circuits

X. Ma and F. Lombardi

Defect characterization of various QCA devices has been presented previously in Section 5.2. In this chapter, fault tolerance of reversible QCA circuit is investigated.

Reversible computing [1] [2] was introduced previously in Section 11.1. Reversible computing entails *virtually* no dissipation scenario in the operation of a system. Under this paradigm, reversible logic bypasses the dissipation lower bound of $2k_BT$ (k_B is the Boltzmann's constant and T is the operating temperature) by avoiding any information loss in the computing [3]. Quantitative evaluation and calculation based on quantum dynamics [4] have shown that QCA can provide the potential implementation for reversible logic.

The manufacturing process for QCA, like other nano-scale technologies, suffers a high fault rate. To assemble a reliable computing system with QCA, fault tolerance and high performance must be utilized in a symbiotic arrangement. Traditional fault tolerant schemes for VLSI are not fully adequate to handle the expected fault rates of QCA. A novel fault tolerant scheme referred to as *majority multiplexing* has been proposed in [5]; it combines the NAND-multiplexing scheme originally proposed in [6] with a 3-input majority voter (MV) to provide good fault tolerant capabilities. However [5] did not provide the bound of tolerable fault rate of the computing modules in the Maj-MUX system. In addition, restoration speed was not considered in [5]. Fault tolerance is readily present in QCA because the MV is the basic device construct for designing QCA circuits. The MV with three inputs requires only five QCA cells.

The objective of this chapter is to find a suitable fault tolerance scheme for QCA. Analysis and comparison of systems using different fault tolerance schemes, such as Triple Modular Redundancy (TMR), NAND-multiplexing, and Maj-MUX, has been presented. Since MV is the basic device construct in QCA, fault tolerance by majority voting is especially suitable for QCA. Maj-MUX for a QCA implementation has been investigated in detail in this chaper. Fault probability improvement and signal restoration speed have been reported for Maj-MUX. It has been shown that the Maj-MUX can tolerate a higher fault rate and restore signals at lower overhead compared to NAND-multiplexing. This chapter also shows that the energy dissipation due to fault correction in majority multiplexing affects the dissipation of the overall QCA reversible circuit. This chapter begins with a review of available fault tolerant techniques. The performance of majority multiplexing technique is then presented and compared with NAND-multiplexing. In addition the energy dissipation related with fault tolerance in the reversible QCA circuit with majority multiplexing technique is discussed.

12.1 HARDWARE REDUNDANCY TECHNIQUES

Different types of redundant scheme have been proposed and used for VLSI; they are also applicable to QCA.

Triple Module Redundant (TMR) is a widely used fault tolerant technique. A TMR system (Figure 12.1) consists of three modules, each of the modules computes the required function. Three copies of the input signals are generated and sent to a module. The outputs of the three modules are then sent to the majority voters (MVs). TMR generates a correct result at the output when no more than one module is faulty. TMRs can be cascaded to further improve the system's reliability. Signal *reliability* is defined to be the probability of the signal being fault free or correct. If the MVs are assumed to be fault free, then every stage in the cascaded TMR system can improve the signal reliability as $R_{out} = (R_{in})^3 + 3(R_{in})^2(1 - R_{in})$ where R_{in} is the reliability of the input signal and R_{out} is the reliability of the output signal. The reliability of outputs of the TMR stage (R_{out}) is higher than the reliability of the inputs (R_{in}) when $R_{in} > 50\%$. If this is extended to the N-Module-Redundant (NMR) scheme, then the reliability is improved as $R_{out} = \sum_{i=0}^{(N-1)/2} \binom{N}{i} (R_{in})^{N-i}(1 - R_{in})^i$, when $R_{in} > 50\%$. The TMR scheme is advantageous for QCA because the basic QCA device is also the 3-input MV.

The reliability of a TMR system with a non-perfect MV is $R_{sys} = R_{MV} \times [(R_m)^3 + 3(R_m)^2(1 - R_m)]$, where R_{sys} is the system reliability, R_m is the

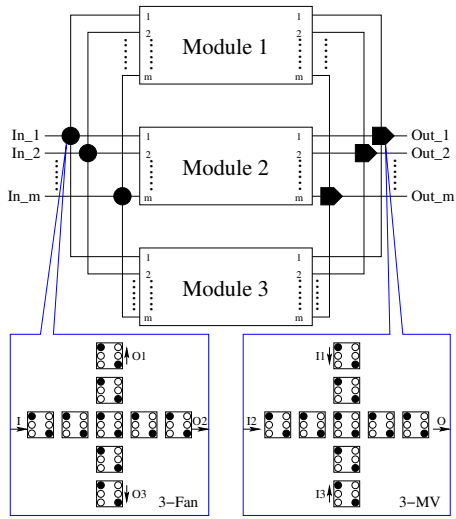


Figure 12.1 A TMR System in QCA

reliability of a module (where R_{MV} is the reliability of the non-perfect MV). For an improvement in system reliability due to TMR, it is required that:

$$R_{sys} = R_{MV} \times [(R_m)^3 + 3(R_m)^2(1 - R_m)] > R_m \tag{12.1}$$

$$\Longrightarrow$$

$$R_{m,a} < R_m < R_{m,b} \tag{12.2}$$

$$R_{m,a} = \frac{3R_{MV} - \sqrt{9R_{MV}^2 - 8R_{MV}}}{4R_{MV}}$$

$$R_{m,b} = \frac{3R_{MV} + \sqrt{9R_{MV}^2 - 8R_{MV}}}{4R_{MV}}$$

If $R_{MV} < 8/9 \approx 0.8889$, then (12.1) can not be satisfied. So, R_{MV} must be $> \frac{8}{9} \approx 0.8889$ to improve over a module reliability. Also, R_m must be greater than $R_{m,a}$ to utilize the fault tolerant capability of TMR. If the reliability of a module is too low, a concatenated TMR system (Figure 12.2) can be employed.

By dividing a large module into serially connected stages, the reliability of each stage (denoted by R_{m_i}, for stage i) becomes suitable for a TMR scheme. The

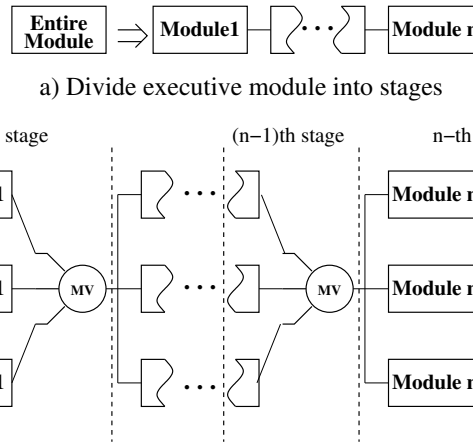


Figure 12.2 Concatenated TMR System

reliability of a system with n stages is

$$R_{sys1} = \prod_{i=1}^{n} R_{MV} \times [R_{m\_i}^3 + 3R_{m\_i}^2(1 - R_{m\_i})]$$

So, this reliability is limited by the reliability of the MVs (i.e., R_{MV}). To avoid the reliability bottleneck represented by R_{MV} for system reliability, a TMR system can be modified as shown in Figure 12.3. The outputs of each stage (as shown in Figure 12.3) must be restored. For the outputs of the three modules at the kth stage, the probability to produce a restored result is

$$[R_{m\_1}^3+3R_{m\_1}^2(1-R_{m\_1})]\times\prod_{i=2}^{k}[(R_{m\_i}\times R_{MV})^3+3(R_{m\_i}\times R_{MV}))^2(1-R_{m\_i}\times R_{MV})]$$

So, the reliability of this n-stage system is

$$\begin{aligned} R_{sys2} \quad = \quad & [R_{m\_1}^3 + 3R_{m\_1}^2(1 - R_{m\_1})] \\ & \times \prod_{i=2}^{n}[(R_{m\_i} \times R_{MV})^3 + 3(R_{m\_i} \times R_{MV}))^2(1 - R_{m\_i} \times R_{MV})] \\ & \times R_{MV} \end{aligned}$$

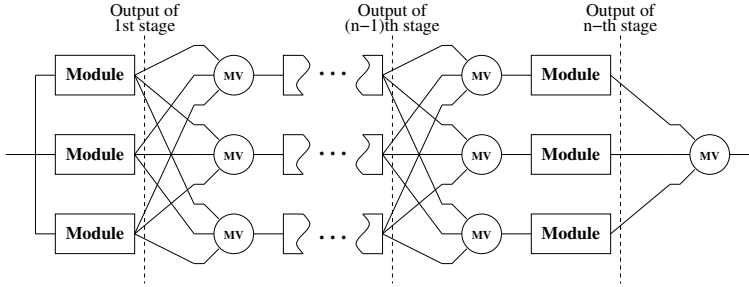


Figure 12.3 A TMR system with MV Redundancy

The reliability of a concatenated TMR system with MV redundancy is higher than a normal concatenated TMR system (i.e., $R_{sys2} > R_{sys1}$) when

$$(R_{m\_i} \times R_{MV})^3 + 3(R_{m\_i} \times R_{MV}))^2(1 - R_{m\_i} \times R_{MV}) > R_{MV} \times [R_{m\_i}^3 + 3R_{m\_i}^2(1 - R_{m\_i})] \quad (12.3)$$

Solving (12.3) gives,

$$R_{m\_i} > \frac{3}{2(1 + R_{MV})}$$

The reliability of the single MV TMR and redundant MV TMR systems are plotted and compared in Figure 12.4. The plots confirm the above calculation, i.e., the redundant MV TMR system has a higher reliability when $R_{m\_i} > \frac{3}{2(1+R_{MV})}$

Dynamic redundancy is used for systems with a high failure rate. A dynamically redundant system can tolerate more faulty modules than an NMR system. For example, a dynamically redundant system with five redundant modules can tolerate up to three faulty modules, while a NMR system with five modules can at most tolerate two faulty modules. However, this technique requires a more complex circuitry than TMR. Thus, it has a higher hardware cost and probability of failure in the fault tolerant circuit.

NAND multiplexing [6] uses NAND gates and random permutation multiplexing to restore a bundle of faulty copies of the same signal. As shown in Figure 12.5, there are N_{bundle} redundant copies of the computing module and its output signal. The unit *U* randomly permutates the signals. The NAND gates are used to restore the signals (i.e., to decrease the failure probability). Although quantitative analysis of NAND multiplexing is difficult, a probabilistic analysis has shown that this technique provides good fault tolerant performance under a high fault rate, albeit a high redundancy rate is needed.

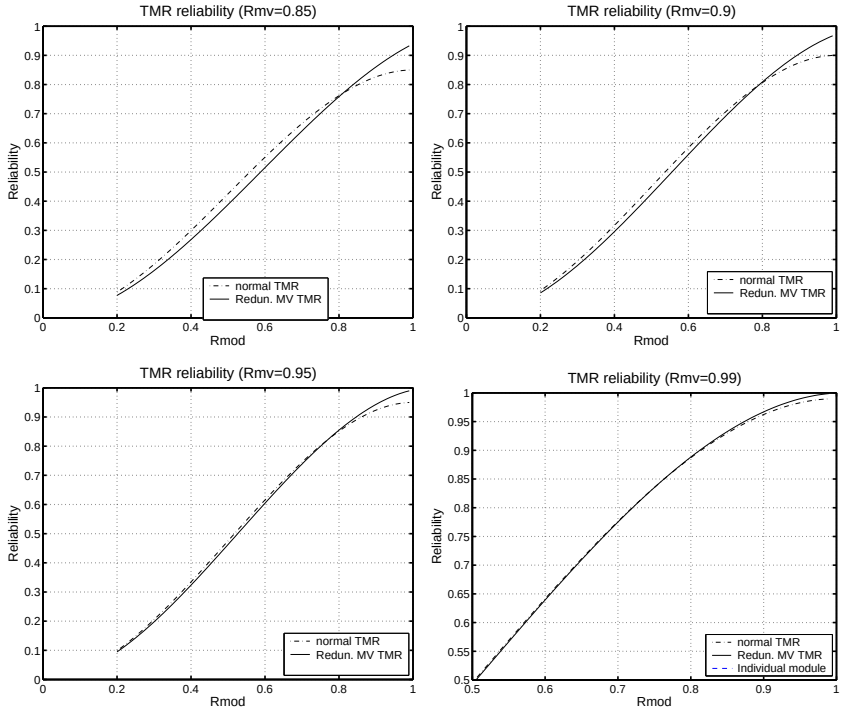


Figure 12.4 Reliability of One Stage in a Concatenated TMR System

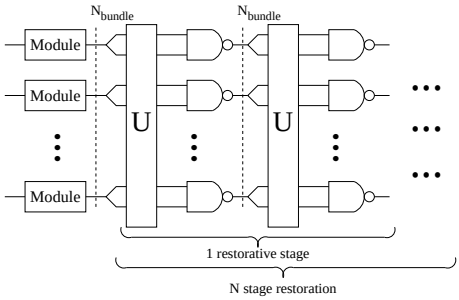


Figure 12.5 A NAND Multiplexing System

An example of a multiplexing system with NAND gates as computing modules was analyzed [6]. It has been shown that with extremely large N_{bundle}, the tolerable fault probability of a NAND gate should be at least 0.0107. A tighter bound on the tolerable fault probability has been pursued by probability analysis. In [7], it has been proved that a NAND gate with a fault rate ϵ smaller than $\epsilon_0 = \frac{(3-\sqrt{7})}{4} \approx 0.08856$ can restore faulty signals from an computing module to a distinguishable level. With multiple levels of restorative stages and a large amount of redundancy, the restored signal fault probability is a function of ϵ only.

12.2 MAJORITY MULTIPLEXING IN QCA

For a QCA system with a high failure rate, a possible approach for establishing the most suitable fault tolerant technique consists of tolerating both high permanent manufacturing and operational (transient) fault rates. Due to the inability of current nanotechnology, the system is likely to be unreliable when manufactured (at time 0), so the treatment of transient faults during time $[0, t]$ has not yet been addressed. In this chapter, the fault probability is analyzed by considering only manufacturing faults.

None of the traditional fault tolerant techniques by themselves can provide a satisfactory solution for QCA. They are either unable to deal with the high fault rate of QCA devices, or unable to have acceptable redundancy costs. The combination of different fault tolerant techniques is therefore a possible solution to achieve a reliable system based on fault-prone QCA devices. For QCA, due to the compact implementation of a majority voter, a cascaded voting scheme is a good basis as a possible fault tolerant solution. Due to its easy QCA implementation and its better capability in restoring signals, the use of a MV in place of a NAND gate in a NAND multiplexing technique is intuitively appropriate (Figure 12.6). This arrangement has been proposed in [5] and is generally referred to as *majority multiplexing* (MajMUX). In this section, the fault tolerant capabilities as well as signal restoration speed have been reported.

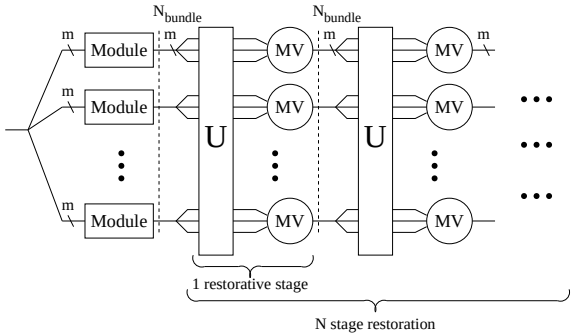


Figure 12.6 A Majority Multiplexing System

12.2.1 Fault Tolerant Capacity

12.2.1.1 Perfect Multiplexing Unit

Using the same method as in [6], the tolerable MV fault rate of a Maj-MUX scheme must be at least 0.0197. [5] Using [7], the tight bound of the fault rate of the MV can be found such that it can be used also in the Maj-MUX scheme.

Assume the inputs of the MVs have an equal fault probability given by x and the fault rate of the MVs is ϵ. Then, the probability $x1$ of the MV outputs being faulty is:

$$\begin{aligned} x_1 &= 1-(1-\epsilon)[(1-x)^3+3x(1-x)^2] \\ &= (2\epsilon-2)x^3+(3-3\epsilon)x^2+\epsilon \end{aligned} \tag{12.4}$$

The worst case scenario is analyzed, so the probability of fault masking are not considered. To have an improved reliability, the condition $x_1 < x$ must hold. Thus,

$$\begin{aligned} (2\epsilon-2)x^3+(3-3\epsilon)x^2+\epsilon &> x \\ (x-1)[2(\epsilon-1)x^2+(1-\epsilon)x-\epsilon] &> 0 \end{aligned}$$

Since $x \leq 1$ and $x = 1$ is not of interest, then

$$2(\epsilon-1)x^2+(1-\epsilon)x-\epsilon < 0$$

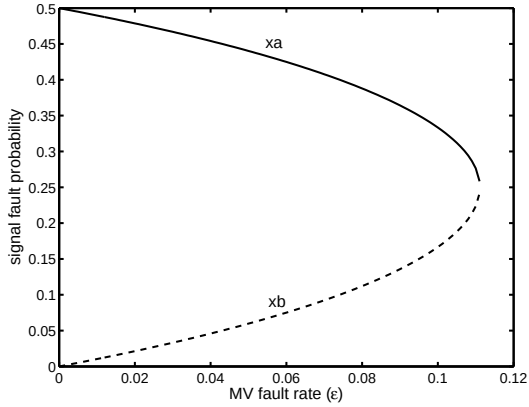


Figure 12.7 Range of Fault Probability Improvement for Maj-MUX

By solving this equation:

$$\begin{aligned} x_a &= \frac{(1-\epsilon)+\sqrt{(9\epsilon-1)(\epsilon-1)}}{4(1-\epsilon)} \\ x_b &= \frac{(1-\epsilon)-\sqrt{(9\epsilon-1)(\epsilon-1)}}{4(1-\epsilon)} \end{aligned}$$

If $1/9 < \epsilon < 1$, then the above equation cannot be satisfied. If $\epsilon \in [0, 1/9]$, for those signals with $x \in [x_b, x_a]$, their fault probability can be decreased to x_b (Figure12.7).

12.2.1.2 Non-Perfect Multiplexing Unit

For a Maj-MUX scheme implemented in QCA, another important source of error is the interconnection, in particular the random multiplexing unit (given by U in Figure 12.8). To improve the reliability of the system, the fault probability of the output signals of a multiplexing unit must be smaller than the previous multiplexing units. The probability that faults in multiplexing unit result in a signal error, is denoted by μ.

$$\begin{aligned} x_1 &= 1-(1-\mu)(1-\epsilon)[(1-x_0)^3+3x_0(1-x_0)^2] \qquad (12.5) \\ x_1 &< x_0 \end{aligned}$$

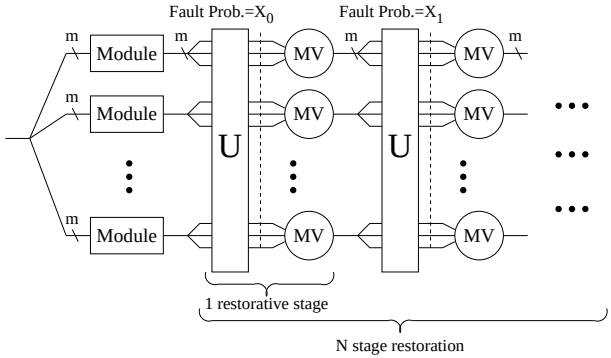


Figure 12.8 Fault in Multiplexing Connection

By substituting $(1-\mu)(1-\epsilon)$ with $(1-\beta)$, the equation above becomes the same as Equation 12.4. So by solving the equation the same result is obtained as

$$\begin{aligned} x_a &= \frac{(1-\beta)+\sqrt{(9\beta-1)(\beta-1)}}{4(1-\beta)} \\ x_b &= \frac{(1-\beta)-\sqrt{(9\beta-1)(\beta-1)}}{4(1-\beta)} \end{aligned}$$

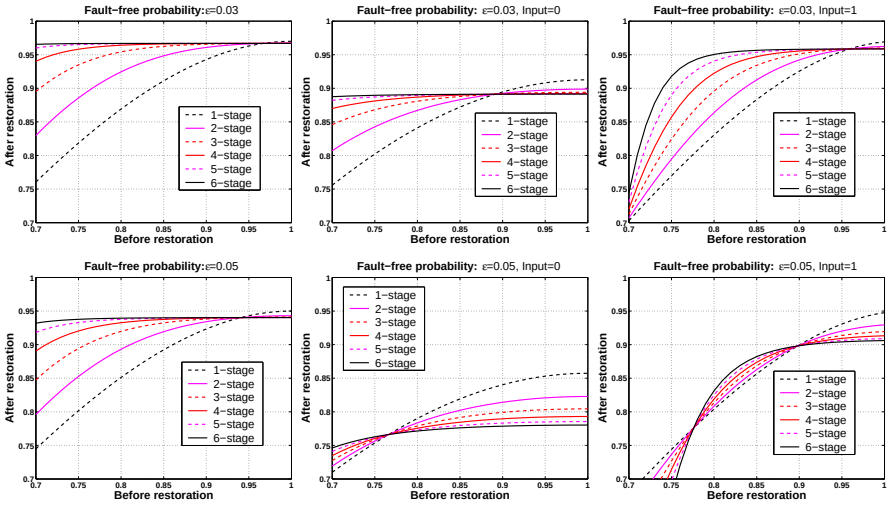
If $1/9 < \beta = (\mu + \epsilon - \mu\epsilon) < 1$, the equation $x_1 < x_0$ cannot be satisfied. The fault probability of the outputs of the computing modules is $x = 1 - \frac{1-x_0}{1-\mu}$. If $\beta \in [0, 1/9]$, for those signals with fault probability $x \in [\frac{x_b-\mu}{1-\mu}, \frac{x_a-\mu}{1-\mu}]$, the fault probability can be decreased to $\frac{x_b-\mu}{1-\mu}$.

12.2.2 Restoration Speed of Multiplexing

Restoration speed is defined as the fault probability improvement that can be achieved with one restorative stage. It is a figure of merit that establishes the number of restorative stages that are needed to assemble a reliable system.

For a NAND multiplexing system, the reliability (i.e. the probability of being fault free) of a signal after one restorative stage given by (where x is the probability of the signal being faulty prior to restoration):

$$if\ input = 1:$$



(a) Maj-MUX (b) NAND-MUX (input=0) (c) NAND-MUX (input=1)

Figure 12.9 Comparison of Restoration Speed for Maj-MUX and NAND-MUX

$$P[ff\ after\ 1\ NAND] = (1-\epsilon)(1-x)^2 \quad (12.6)$$

$$P[ff\ after\ 1\ stage] = (1-\epsilon)(2-P[ff\ after\ 1\ NAND]) \times P[ff\ after\ 1\ NAND] \quad (12.7)$$

$$if\ input = 0:$$

$$P[ff\ after\ 1\ NAND] = (1-\epsilon)(1-x^2) \quad (12.8)$$

$$P[ff\ after\ 1\ stage] = (1-\epsilon)P[ff\ after\ 1\ NAND]^2 \quad (12.9)$$

For a Maj-MUX system, the faulty probability after one restorative stage is given by

$$P[ff\ after\ 1\ stage] = (1-\epsilon)[(1-x)^3 + 3x(1-x)^2] \quad (12.10)$$

Figure 12.9 shows the signal reliability (as the probability of a signal being fault free) after different numbers of restorative stages. The NAND multiplexing and Maj-MUX schemes are compared under different values of ϵ. The Maj-MUX scheme has a faster signal restoration speed than the NAND-MUX scheme. For example, with the error rate of MV and NAND both at $\epsilon = 0.03$ and signal

reliability before restoration $= 0.8$, Maj-MUX needs 4 restorative stages to recover the signal to get a full fault tolerance, while NAND-MUX needs 6 stages.

12.2.3 Summary

It has been shown that the Maj-MUX scheme is attractive for QCA because of the following:

- A Maj-MUX scheme requires a lower reliability for the majority voter (0.8889) than NAND multiplexing for the NAND gate (0.91144). In QCA, a MV has a every compact implementation. As a MV requires only five QCA cells, then it is possible to reach the gate reliability requirement of the Maj-MUX scheme. This advantage makes the Maj-MUX scheme suitable for QCA implementation.
- Given a sufficient number of restorative stages and redundancy rate, the tolerable fault rate of a computing module is very high (for example, the module can be 0.333 faulty if the fault rate of the MV is 0.1). This fault tolerant bound and final fault probability of restored signals are set by the reliability of the restorative stages. The restored signal reliability is $\frac{(1-\epsilon)-\sqrt{(9\epsilon-1)(\epsilon-1)}}{4(1-\epsilon)}$. Using Taylor expansion, the reliability is $\epsilon+3\epsilon^2+O(\epsilon^3)$. For $\epsilon < 0.1$, the restored signal reliability is approximately ϵ, as the reliability of MV.
- As shown in Figure 12.9, the Maj-MUX scheme has a better restoration speed than NAND-MUX.

However, the following disadvantages are incurred using the Maj-MUX.

- The redundancy rate considered in this work is very large. Ultimately, the fault tolerant capability of this scheme will be limited by the redundancy rate that the system can afford.
- An implementation of Maj-MUX will require a large amount of wire crossing devices in QCA. The reliable operation of the wire crossing device is therefore crucial for assessing the applicability of this fault tolerant scheme.
- A multiplexing scheme (using a MV or a NAND gate) can preserve a high reliability of a system, however its output signals are provided in bundles. So, for a traditional output signal, there will be a threshold (or voted) logic to

reduce the bundle-signal to a bit-signal. The reliability of these "final" output gates may affect the system reliability.

12.3 REVERSIBLE COMPUTING AND FAULT TOLERANCE

Reversible computing is one of the possible solutions to the energy hurdle that prevents the increase of integration density in computing systems. In Chapter 11, the conditions for a QCA cell to operate reversibly have been presented. In a clock cycle of operation, if a QCA cell is released under a driver with same polarization, the whole operation cycle is reversible. Otherwise, if there is no driver during the "release" phase, at least k_BT energy will be dissipated; if it is released under a driver with opposite polarization, at least $2k_BT$ will be dissipated.

Using the mechanical model introduced previously in Chapter 11, misplacement defects are considered and fault characterization is established at logic level. The static state energy of different circuit units such as MV and coplanar crossing interconnection is calculated. Under different single cell displacement defects, the energy of fault free and faulty output logic value are compared. The comparison gives the condition under which a single cell displacement will result in logic fault.

Consider the MV, shown in Figure 12.10, as an example. The cell misplacement from the original position is denoted by d. Assume the inputs of the MV are $ABC = 101$. The four types of misplacement studied using the mechanical model are illustrated in Figure 12.10.

1. The first type of defect considered in this chapter is the misplacement of the top input A in the *y* direction (i.e., A is moved north). When the misplacement $d > 0.67a$, an output with erroneous value appears.
2. For the misplacement of the top input A in the *x* direction, the MV functions correctly for $-0.87a < d < 0.65a$.
3. For the *x* direction misplacement of input B, the correct value is always present. However the difference between the energy of the correct and erroneous states decreases by increasing d.
4. For the *y* direction misplacement of B, the erroneous MV function is caused by a defect with misplacement $d < -1.21a$ or $d > 1.21a$.

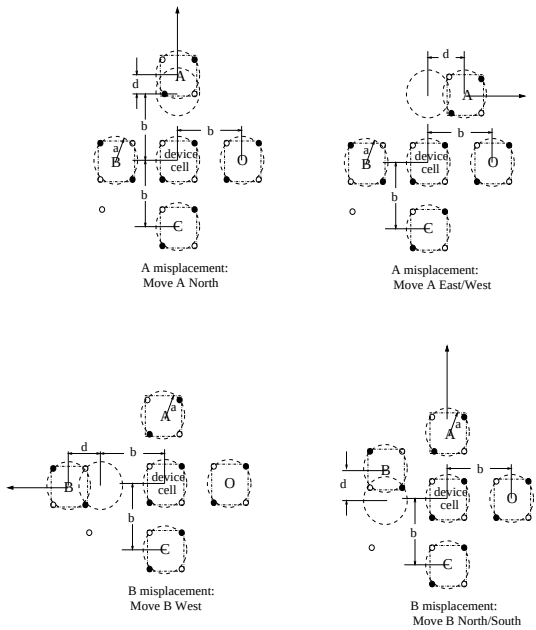


Figure 12.10 Cell Misplacement in MV

12.4 ENERGY DISSIPATION OF A REVERSIBLE MV MULTIPLEXING SYSTEM

In the majority multiplexing scheme, if each module is reversible, then reversibility could also be accomplished at system level. The random permutation multiplexing is a reversible function, so it can always be implemented with a reversible logic circuitry. An MV for TMR has three inputs and one output. Its function is not logically reversible. If there is no fault in the circuit, the use of this majority voter in a TMR system does not entail dissipation into the reversible circuit (see Chapter 11). If a fault exists in a system, then there are two sources of possible dissipation: the circuits that produce the fault and the fault tolerant circuits that mask the fault.

The computing module is designed to be reversible irrespective of its input pattern. So a fault-free module will not dissipate energy when receiving faulty inputs. Dissipation in the faulty module is not caused by the fault tolerant circuitry, and therefore outside of the scope of this chapter. This chapter concentrates on the dissipation related with fault masking.

12.4.1 System Without Fault

It has been shown previously in Chapter 11 that fanout and MV of QCA do not necessarily dissipate energy under unanimous inputs. So they can be included in reversible computing circuits. Assume that in a system E_{bit} energy is used to encode one bit of information. In order to be distinguished from thermal noise, E_{bit} must be at least $k_B T$. For a 1-to-n fanout (denoted by n-*fan*), $(n-1) \times E_{bit}$ needs to be injected into the circuit to encode the extra $n-1$ copies of information bit. For an n-input MV (denoted by n-*MV*), when all the n inputs are the same, $(n-1) \times E_{bit}$ will be sent back to the energy source. In QCA, the clocking system is the energy source. In both cases, there is no lower bound of energy dissipation. So, if a 3-fan and a 3-MV are connected together (Figure 12.11), the energy absorbed by the fanout can be sent back to the energy source when the n copies go through the majority voter and are reduced to just 1 copy.

For example, a reversible TMR system has three reversible circuit modules. The number of input and output signals of a reversible circuit are the same, and this number is denoted by m. m 3-fan fanout structures are employed to send copies of input signals to the three modules. m 3-MVs are used to generate final outputs from modules' outputs. If there is no fault in the system, every 3-fan and 3-MV works as described above. So the whole system remains reversible. Energy sources provide

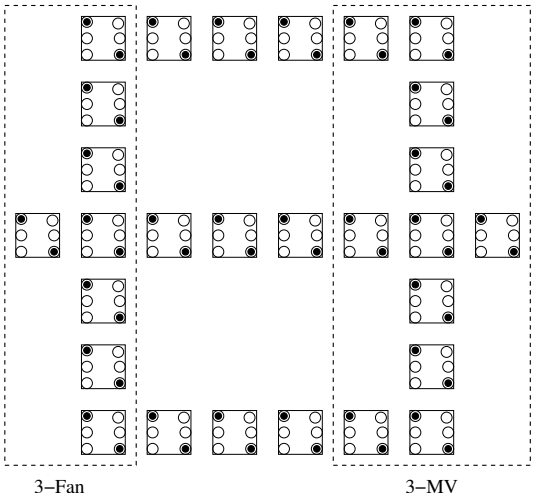


Figure 12.11 A circuit with 3-Fan and 3-MV connected together

$2m \times E_{bit}$ energy for the m 3-fan unit, then get $2m \times E_{bit}$ back from the m 3-MV. No dissipation occurs in the fault tolerance circuit (see Chapter 11 for detail).

The above analysis can be applied to the majority multiplexing system shown in Figure 12.12. Assume there are nine computing modules in the system. There are $4m$ 3-fan fanout structures to copy primary input signals to the modules. $18m$ 3-fan fanouts and $18m$ 3-MVs are used in the restorative stages, and $4m$ 3-MVs are used to generate final output. Energy source provides $44m \times E_{bit}$ energy for the $22m$ 3-fan unit, then get $44m \times E_{bit}$ back from the $22m$ 3-MV. No dissipation occurs in the fault tolerance circuit.

12.4.2 Dissipation in Fault Correction

Although a fault free reversible majority multiplexing system can have energy dissipation infinitely close to zero, correcting faulty signals will cause energy dissipation.

12.4.2.1 System with Faulty Computing Modules

First assume the MV and the multiplexing unit are fault free. If the inputs of one 3-MV are different, $2E_{bit}$ of dissipation will occur for every minority input.

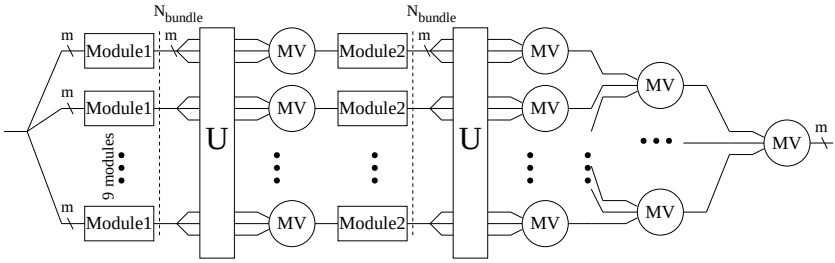


Figure 12.12 Example of Majority Multiplexing System

E_{bit} has been defined in section 12.4.1. For a restoration stage, its dissipation is generated by MVs with either 1 or 2 faulty inputs. Every one of such MVs dissipates $2E_{bit}$. Given the error rate ϵ_{sig} of its input signals, dissipation of the stage is $E_D = [3\epsilon_{sig}(1-\epsilon_{sig})^2 + 3\epsilon_{sig}^2(1-\epsilon_{sig})] \times 2E_{bit}$. So, for a n-stage restoration, the dissipation is

$$6E_{bit} \times M \times \sum_{k=1}^{n} [\epsilon_{sig\_k} - \epsilon_{sig\_k}^2] \tag{12.11}$$

where M is the total number of faulty signals that have been restored and ϵ_{sig_k} is the input signal error rate of the k-th stage and ϵ_{sig_1} is the error rate of initial signal. As shown in Section 12.2 , ϵ_{sig_k} can be calculated iteratively as:

$$\epsilon_{sig\_k} = [\epsilon_{sig\_k-1}^3 + 3(1-\epsilon_{sig\_k-1})\epsilon_{sig\_k-1}^2] \tag{12.12}$$

12.4.2.2 System with Faulty MVs and Faulty Computing Modules

The fault in majority voter is considered in addition to the faulty signals from computing modules. Because of the logical fault in MV, the signal error rate of each restoration stage is higher than that predicted by Equation 12.12. As shown in Equation 12.4, the signal error rate with faulty MVs is

$$\epsilon_{sig\_k} = 1 - (1-\epsilon) \times [(1-\epsilon_{sig\_k-1})^3 + 3(1-\epsilon_{sig\_k-1})^2\epsilon_{sig\_k-1}] \tag{12.13}$$

where ϵ is the logical fault rate of MV. Since only dissipation in correcting the error is considered, only fault free MV is included in this dissipation calculation. The dissipation of an n-stage restoration is

$$6E_{bit} \times M \times \sum_{k=1}^{n} \{(1-\epsilon)[\epsilon_{sig\_k} - \epsilon_{sig\_k}^2]\} \tag{12.14}$$

12.4.2.3 System with Faulty MVs, Faulty Computing Modules and Faulty Multiplexing Units

Considering the fault in multiplexing unit, the signal error rate of each restoration stage can be derived from Equation 12.5:

$$\epsilon_{sig\_k} = 1 - (1-\beta)(1-\epsilon) \times [(1-\epsilon_{sig\_k-1})^3 + 3(1-\epsilon_{sig\_k-1})^2 \epsilon_{sig\_k-1}] \tag{12.15}$$

where β is the logical fault rate of multiplexing units. The input error rate of MVs in the restoration stages is $1-(1-\beta)(1-\epsilon_{sig\_k-1})$. So the total dissipation from the fault correction of an n-stage restoration is

$$6E_{bit} \times M \times \sum_{k=1}^{n} \{(1-\epsilon)[(1-\beta)(1-\epsilon_{sig\_k}) - (1-\beta)^2(1-\epsilon_{sig\_k})^2]\} \tag{12.16}$$

12.4.2.4 Summary

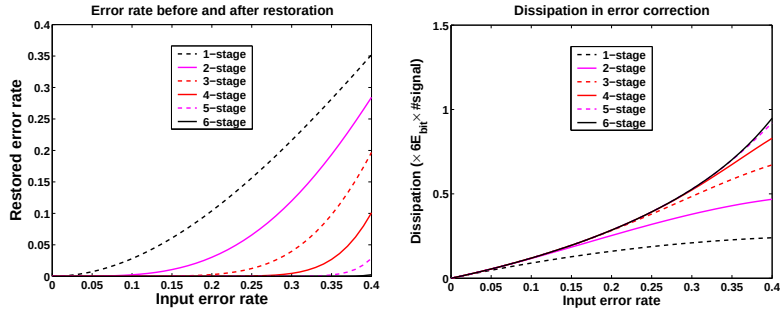
The output signal error rate of every restorative stage is decided by two factors: the input signal error rate and the reliability of the error correction system.

As plotted in Figure 12.13, the output signal error rate and dissipation of Maj-MUX with different restorative stage is shown as an example, under the fault assumptions given above. Figure 12.14(a) and (b) plots output restored error rate and dissipation vs. input error rate for different MV error rate ϵ, respectively. Figure 12.14(c) and (d) plots output restored error rate and dissipation vs. input error rate for different MV error rate ϵ, respectively.

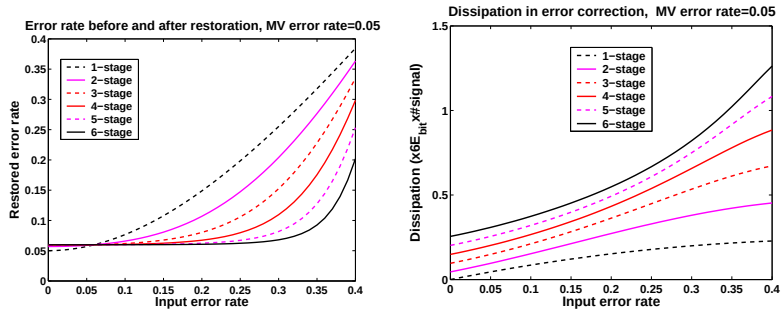
12.5 CONCLUSION

In this chapter, defect tolerance of reversible QCA circuits have been pursued in detail. The fault tolerant capacity and signal recovery speed of Maj-MUX technology have been investigated. It has been shown that this technology can improve system reliability as long as the reliability of the majority voter is higher than 0.8889. In comparison with the NAND-multiplexing technology, Maj-MUX has not only better fault tolerant capacity but also higher signal restoration speed. In addition, the compact implementation of MV in QCA makes Maj-MUX technology especially suitable for QCA circuits.

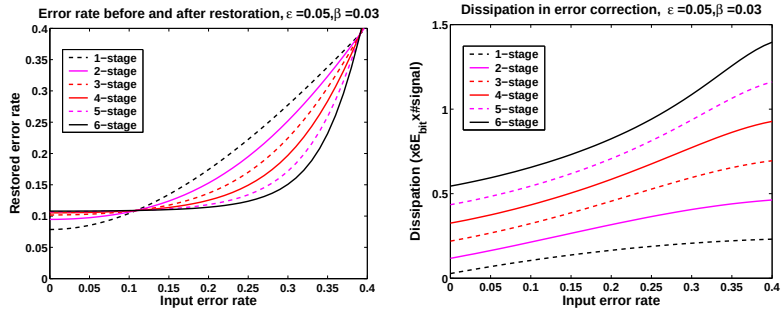
The energy dissipation in QCA reversible circuits that is caused by Maj-MUX technology is analyzed. In the fault free case, the fault tolerant circuitry does not



a) Fault : module only



b) Fault : module + MV



c) Fault : module + MV + Mux. unit

Figure 12.13 Error and Dissipation in Restorations of Different Stage Number

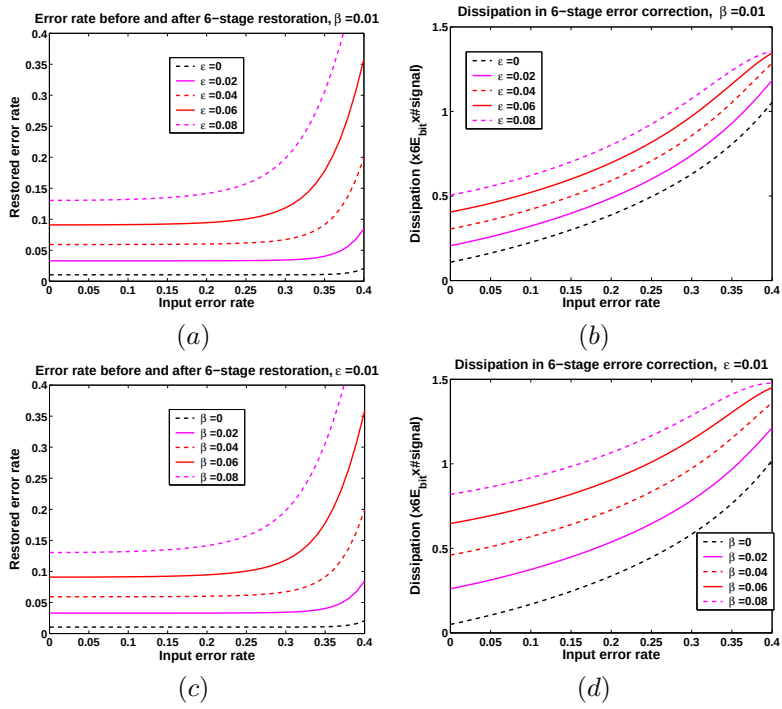


Figure 12.14 Error and Dissipation of 6-stage Restoration

cause extra dissipation. When faults occur, the energy dissipation caused by fault correction has been derived from error rates of different parts of the circuit. To the best of our knowledge, our work is the first to investigate the energy dissipation caused by fault tolerance in reversible circuit.

The defects of QCA cell, whether they result in logic fault or not, can change the energy dissipation of QCA circuit. Our current work focuses on the energy dissipated to correct logic fault. In order to fully understand the energy dissipation issue in faulty reversible QCA circuits, the dissipation directly generated by defects should also be characterized.

Current work uses the fault rate of circuit units to characterize the reliability of system and the dissipation. Gate-level fault modelling need to be performed to give the circuit unit fault rate in terms of cell defect rate.

References

[1] Bennett, C. H. “Logic Reversibilty of Computation”, *IBM Journal of Research and Development*, vol 17, 1973, pp. 525-532.

[2] Toffoli, T., “Reversible Computing”, *Technical Report MITLCSTM151, MIT Laboratory for Computer Science*, 1980.

[3] Landauer, R., “Irreversibility and Heat Generation in the Computing Process”, *IBM Journal of Research and Development*, vol 5, 1961, pp. 183-191.

[4] Lent, C.S., M. Liu, and Y. Lu, “Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling,” *Nanotechnology*,Vol. 17, No. 16, 2006, pp. 4240-4251.

[5] Sandip, R. and V. Beiu, “Majority Multiplexing - Economical Redundant Fault-Tolerant Design for Nano Architectures”, *IEEE Transcation on Nanotechnology*,Vol. 4, No. 4, 2005, pp. 441-451.

[6] von Neumann, J., “Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components,” in *Automata Studies*, pp. 43-98, C. E. Shannon, and J. McCarthy (Eds.), Princeton, NJ: Princeton Univ. Press, Princeton, NJ, 1956.

[7] Evans, W. and N. Pippenger, “On the Maximum Tolerable Noise for Reliable Computation by Formulas”, *IEEE Transaction on Information Theory*, Vol. 44, No. 3, 1998, pp. 1299-1305.

Chapter 13

Conclusion and Future Work

F. Lombardi

QCA is a promising alternative technology to CMOS. It has been anticipated that QCA implemented with molecules will provide room temperature operation and improvement in speed, density and power consumption over existing CMOS systems. In this book, various designs as well as defect tolerance aspects of QCA are analyzed. As an emerging technology, QCA is radically different from CMOS, which calls for different design, logic synthesis and fault tolerant techniques.

QCA provides a new method of computation in which information is presented as charge configurations of electrons within confined sets of quantum dots. Recent research indicates QCA systems can be manufactured using molecular self-assembly where each QCA cell is a single molecule. These bottom-up fabrication mechanisms are likely to have defect rates that are orders of magnitude higher than traditional CMOS. Consequently, investigating defect and fault tolerance methodologies and design principles suitable for such highly unreliable architectures becomes certainly necessary and important. In this book, we have characterized various defects and failure mechanisms in both combinational and sequential QCA devices and circuits. It has been shown that QCA defects in many cases result in unwanted inversion faults at logic level. The device fault characterization is utilized for generating test vectors through a weighted (grading) approach. A heuristic approach based on a novel metric is proposed as criterion for selecting and prioritizing vectors when testing a QCA circuit.

New design methodologies have been proposed for QCA in this book. Both combinational design as well as sequential design have been considered. In QCA, the basic logic gate is the Majority Voter (MV). It has been shown that existing logic

synthesis tool does not make efficient use of the MV. This indicates a need for new majority logic based synthesis algorithm for QCA technology. QCA is a technology where information transformation and computation occur simultaneously. Clocking in QCA not only provides signal gain but also enforced pipelining. For QCA, even the combination circuits are clocked and therefore pipelined. A novel two-dimensional clocking scheme has been proposed in this book, which permits a reduction in the longest line length in each clocking zone. This reduction permits a fast timing and efficient pipelining to occur, while guaranteeing kink-free behavior in switching. Sequential design poses unique challenges in QCA. As the basic units in sequential design, flip-flops of both D-type and RS-type have been introduced in this book. Because of the timing constraints imposed by clocking, it is required that all paths from flip-flop to flip-flop must have the same delay (the number of clocking zones). An algorithm that assigns clocking zones and stretches paths to match delay when required has been proposed.

A tile-based modular design methodology has been introduced in this book. The design is based on basic building blocks referred to as tiles. The tile-based design has been shown to offer versatile logic operation and is defect tolerant. The modular design is also well suited for molecular QCA where it is anticipated that manufacturing will be done by large scale cell deposition. Additionally, modular design can be implemented within a CAD framework. Combinational as well as sequential designs using the tile-based methodology have been demonstrated. A serial memory architecture based on tiles has been proposed in this book. The QCA paradigm of memory-in-motion has been accomplished using a novel arrangement in the storage loop. A three-zone memory tile is used.

Reversible computing with QCA has been investigated in this book. A mechanical model for QCA cells has been proposed, which can be used in analyzing the energy dissipation and reversibility of QCA circuits with different clocking schemes. System level defect tolerance for reversible QCA circuits has been pursued. Fault tolerance capacity, signal restoration speed and energy dissipation has been investigated for QCA circuits using the Maj-MUX scheme.

As with many other emerging technologies, QCA faces many challenges. New manufacturing techniques must be developed such that large scale QCA circuits can be made economically and reliably. Interface between QCA and CMOS is required so that signal can be applied and read from a QCA circuit. A CAD framework, including logic synthesis, place and route, as well as automatic layout generation, needs to be built for QCA. As self-assembly processes mostly likely have high defect rates, investigating defect and fault tolerance methodologies and

design principles suitable for such highly unreliable architectures becomes certainly necessary and important.

Appendix A

Preliminary for QCA Mechanical Model

X. Ma and F. Lombardi

The relation between computing and energy dissipation was initially investigated by Landauer who showed that k_BTln2 joules of energy are generated for each bit of information lost due to the non-reversibility in the computational process [1] (where k_B is Boltzmann's constant and T is the operating temperature). Moreover, if computation is performed in a reversible manner, it has been shown that k_BTln2 energy dissipation would not necessarily occur. In reversible computing, the input state of a device can always be uniquely established from its output state (one-to-one onto mapping). This avoids the irreversible process in computation, thus making it possible at least in theory to build computational systems whose energy dissipation is only determined by the number of inputs and outputs, not by the number of gates in the system. For a large system, the amount of energy per gate can be made very small, so that the high density integration of systems manufactured in the nano-scale will not be limited by energy dissipation. To investigate the reversibility and energy dissipation QCA circuit, an analysis of general computing system is first pursued.

In a computing system, the degrees of freedom of its components are encoded to bear information. A thermodynamic model of a single ideal-gas molecule is presented. Its operation is analyzed in terms of information, entropy and dissipation. By analogy, the relation between entropy change and heat dissipation derived in this model is used in the analysis of the operation and dissipation of the computing model proposed in Chapter 11.

System entropy increases during loss (or destruction) of information. According to thermodynamics [2], the change of system entropy is $\Delta S = k \ln \frac{W_f}{W_i}$, where k_B is Boltzmann's constant, W_i and W_f are the number of possible sub-states in

the initial and final states, respectively, Q is the heat that the system absorbs from the environment, and W is the work done by the system.

For example, an ideal gas has six degrees of freedom (three dimensions of space position and three directions of momentum). If a gas with N_A molecules changes its volume from V_i to $V_f = 2V_i$ in an isothermal expansion, then the change of its entropy is given by $\Delta S = k \ln(V_f^{N_A}/V_i^{N_A}) = N_A k \ln(2)$. Isothermal expansion is reversible [2], so this increase in entropy comes from the heat absorbed from the environment and $Q = \Delta S \times T = N_A \times k_B T \ln 2$ (Q is positive when the heat goes from the environment to the system). The internal energy of a gas is constant in an isothermal expansion, so the work $W = -Q$ is done to the gas (W is positive when work is done to the system). If the change in volume is achieved by free expansion, then there is no work done in the process (i.e., $W = 0$). The internal energy of the system does not change. So, there is no heat exchange between the system and the environment, $Q = \Delta E_{internal} - W = 0$. Free expansion is not reversible, so the change of entropy ΔS is larger than $\int \frac{dQ}{T} = \frac{Q}{T} = 0$.

In a computing system, some degrees of freedom can be used to encode information. So, with no loss of generality, a bi-state computing unit divides all possible states into two sub-spaces, according to the information-bearing degrees of freedom. Consider again the example of an ideal gas; if a gas molecule in a cell (container) with volume $2V$ is utilized as a bi-state unit, then information can be encoded by defining a first state as 1 if the molecule is in the upper half of the cell, and a second state as 0 if the molecule is in the lower half of the cell (shown in Figure A.1). If there is no separation, the gas can move freely in the cell, thus changing the cell state between 1 and 0. Entropy in this free state is denoted by S_0. The state can be set to 1 by moving the bottom to the middle of the cell. Similarly, moving to the top can set the cell to the 0 state. Assume the movement is slow enough to keep the operation isothermal, this operation places one bit of information into the cell, and the entropy of the cell becomes $S_0 - k \ln 2$. If the temperature of the system is denoted by T, then $W = k_B T \ln 2$ of work is done to the cell during the operation, and the heat exchange is given by $Q = -k_B T \ln 2$.

By knowing the state of a cell, it is then possible to change it from 1 or 0 to the free expansion state by moving the separating wall to the corresponding position (bottom or top). This operation increases the entropy back to S_0; also, $W = -k_B T \ln 2$ and $Q = k_B T \ln 2$ are needed for this operation. In the cycle of this process (often referred to as set-then-erase), the total work and heat dissipation are both 0. This is an erasure with no dissipation and can only be performed when the cell state is known. Consider all the parts involved in this operation as a system; no information is destroyed in this system. Information will be erased if

the separation wall is broken. A molecule's free expansion through the broken wall performs zero work, and the internal energy of the gas experiences no change. So, there is no heat exchange between the system and the environment. Meanwhile, the cell entropy increases to S_0 during free expansion. For the working cycle of the set-then-erase operation through free expansion, the work $W = k_B T \ln 2$ must be done to the system and $Q = -k_B T \ln 2$ is transferred between the gas (as computing system) and the environment (a negative value means that there is heat dissipation from the system).

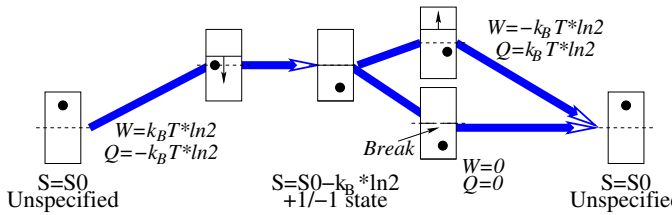


Figure A.1 A Memory Cell of a Gas Molecule

This example illustrates that loss of information entails dissipation. Storing information into a logic cell requires heat flow into the environment to decrease the entropy of the cell from the unspecified state. To recover the cell from the unspecified state, it is possible to absorb heat from the environment. The lower limit for the heat generated in the former process is the same as the upper limit of the heat absorbed in the latter process. Both limits are achieved only by a quasi-equilibrium process. The key element of Landauer's claim is that no quasi-equilibrium process can be applied without knowing the state of the information in the system. However, knowledge of information means that the information erased in the cell is not the only copy in the entire computing system, i.e., the information is not destroyed. If no other copy of the information exists in the computing system, then the above example suggests that the cell can only be recovered to the specified state by a process like free expansion. This process does not absorb heat and the heat dissipation that occurs in the information storage process is the dissipation of the full work-cycle.

In the above discussion, the base of the logarithm was given by e. The selection of the logarithm base does not change the applicable physical laws that the formula uses. As in the remainder of the book, a bi-state system is assumed, so a base of 2 will be used to simplify notation and presentation (albeit, also in this case the notation has no implication on the general validity of the presented analysis).

References

[1] Landauer, R., “Irreversibility and Heat Generation in the Computing Process”, *IBM Journal of Research and Development*, Vol 5, 1961, pp. 183-191.

[2] Fermi, E., *Thermodynamics*, New York, NY: Dover Publications, Inc., 1956

Appendix B

Validation of Mechanical Model

X. Ma and F. Lombardi

B.1 VALIDATION OF STATIC ENERGY ANALYSIS

To verify the validity of the proposed mechanical model, its steady state energy results were compared with those obtained through a computation-based model, such as QCADesigner [3]. Such comparison is valid because the proposed model and QCA both use Coulombic force for the inter-cell interactions and they can both be expressed as electric quadrupoles. So, the same energy states are expected to occur in both of them, as corresponding to equivalent characterizations (e.g., cell size, cell distance and amount of electric charge). So both QCA and the mechanical model should have the same ground state, represent the same logic and compute the same result. In all previous presented cases, the steady state analysis above yields the same result as the simulation result of QCADesigner. This shows that the proposed model is complete as it can be used to characterize the steady state behavior of all QCA circuit primitives, including logic gates and interconnect structures.

Therefore, after computing the energy states of different circuits of mechanical cells, the same circuits have been also assembled with QCA cells and simulated by employing QCADesigner [3]. It has been verified that the simulation results of each and every logic device in QCADesigner are the same as the *ground state* (the state with the lowest energy) of its counterpart version in the proposed mechanical model. As both of these models utilize quasi-adiabatic clocking, then the cells stay in the ground state after switching. The agreement between simulated and computed results further confirms the validity of the proposed model for QCA. Moreover, as

the devices evaluated previously constitute the basic components for building large QCA systems, the proposed model can be utilized with confidence.

B.2 VALIDATION OF DISSIPATION ANALYSIS

In [1] and [2], a quantitative calculation of the operation of several QCA circuits has been presented. The dissipation analysis is made on the same set of circuits as in [1] [2] using the proposed mechanical model.

Erasure of a single cell: [2] has calculated the dissipation of setting and erasing a single cell and reached the conclusion that when utilizing a so-called "Demon cell" with same polarization as the cell being erased, the erasure process has dissipation less than $k_B T \ln 2$. When no such "Demon cell" exists, dissipation is larger than $k_B T \ln 2$. This agrees with the results of Section 11.3.1: at least $k_B T \ln 2$ will be dissipated if a stand-alone cell is erased; however, if a cell of the same polarization is present to drive the cell during the RELEASE phase, then dissipation can be avoided.

Two-cell signal path: Two cells in adjacent clocking zones constitute the simplest circuit under the proposed model (Figure B.1). Over five clocking phases, its operation is as follows:

- Initially, cells 1 and 2 are both in the $NULL$ state, with clocking in the RELAX phase. An external driver is applied to cell 1. With no loss of generality, assume that the driver's value is 1.
- Cell 1 goes through the SWITCH phase. As described in Section 11.3.1, cell 1 has a polarization of 1; the potential energy (E_p) between the driver and cell 1 (denoted as E_d) and the energy between cell 1 and cell 2 (denoted by E_1) are transferred into the clocking unit.
- Cell 1 goes into the LOCK phase and the external driver is removed. Meanwhile, cell 2 acquires the value 1. The potential energy between cell 1 and cell 2 ($E_p = E_2$) is transferred into the clocking unit.
- Cell 1 is placed in the RELEASE phase under the bias of cell 2, that is now in the LOCK phase. As described in Section 11.3.1, cell 1 is under a same polarization condition of bias, so no explicit dissipation occurs; E_2 comes from the clocking unit and becomes potential energy between cell 1 and cell 2.

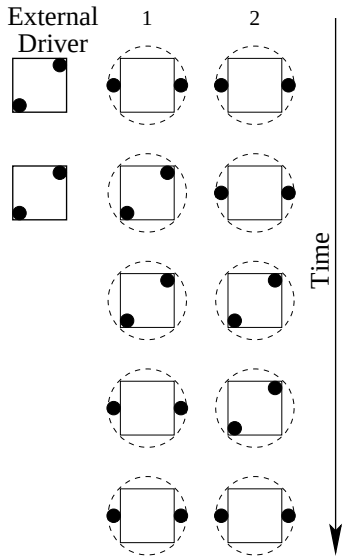


Figure B.1 A Signal Path with Two Cells

- Cell 2 is placed in the RELEASE phase under no bias, so at least Tk of energy is drained from the clocking unit and dissipated. The clocking unit also provides E_1 as potential energy between cell 1 and cell 2.

Over the entire cycle of the circuit, the external driver provides E_d energy. At least k_BT of this energy is dissipated and the remaining energy goes into the clocking unit. A two-cell signal path operates as the "one test cell plus one demon cell" described in [2]. The mechanical model leads to the same conclusion as calculated in [2]: the first cell works reversibly, because the second cell works as a demon cell; the erasure of the second cell is irreversible because when it is released there is no demon cell for it.

Shift register with one cell per stage (SR1): The shift register with one cell per stage (denoted as SR1) can be viewed as the multiple concatenation of two-cell signal paths, as analyzed previously. As illustrated in Figure B.2, cell m receives its logic value from cell $m-1$. When cell $m-1$ is in the RELAX phase, at the same time cell $m+1$ is in the SWITCH phase with the same value of cell m. Then, cell m is in the RELAX phase, while the signal is delivered to cell $m+2$. The distance between the centers of two adjacent cells is denoted by d. When a cell (except for

the first and last cells in the line of the shift register) is in the SWITCH phase, then it is driven by the cell located prior to it. When it is in the RELAX phase, then it is driven by the cell located after it. As proven previously, this behavior of the cells is reversible.

For a shift register with n stages, its operation consists of $n + 2$ phases. All stages (except the last one) work reversibly as described in Section 11.3.1. After passing one bit information through SR1, the circuit receives E_d (as defined in the analysis for a two-cell signal path) from the driver, among which $k_B T$ is dissipated and the rest of the energy goes into the clocking unit. $k_B T$ dissipation is the result of an information loss at cell n. If the output of SR1 is connected to another circuit, then the information propagates into the next circuit and cell n is released under the driving of that circuit. In this case, no dissipation will occur in SR1. As driver, SR1 provides energy to the next circuit, just as it receives energy from its driver. If SR1, its driver and the next circuit have the same design parameters (cell size, distance and charge quantity), then SR1 will provide the next circuit with the same amount of energy of E_d. SR1 is treated as a chain of "demon" cells by [1] [2]; their calculation has confirmed that the energy dissipated per cell per clock switching can be much less than $k_B T \ln 2$.

The dissipation analysis in the proposed model takes into consideration the energy exchange with the clocking system. The clocking system operates like the moving wall in the gas model of Appendix A; it provides or absorbs energy from the computing system during the different phases of the working-cycle, thus making possible to balance the total work in the reversible process.

References

[1] Lent, C. S., M. Liu and Y. Lu, "Bennett Clocking of Quantum-dot Cellular Automata and the Limits to Binary Logic Scaling," *Nanotechnology*,Vol. 17, No. 16, 2006, pp. 4240-4251.

[2] Timler, J. and C. S. Lent, "Maxwell's Demon and Quantum-dot Cellular Automata," *Journal of Applied Physics*, vol 94, no 2, 2003, pp. 1050-1060.

[3] Walus, K., et al., "QCADesigner: A CAD Tool for an Emerging Nano-Technology," *Micronet Annual Workshop*, 2003.

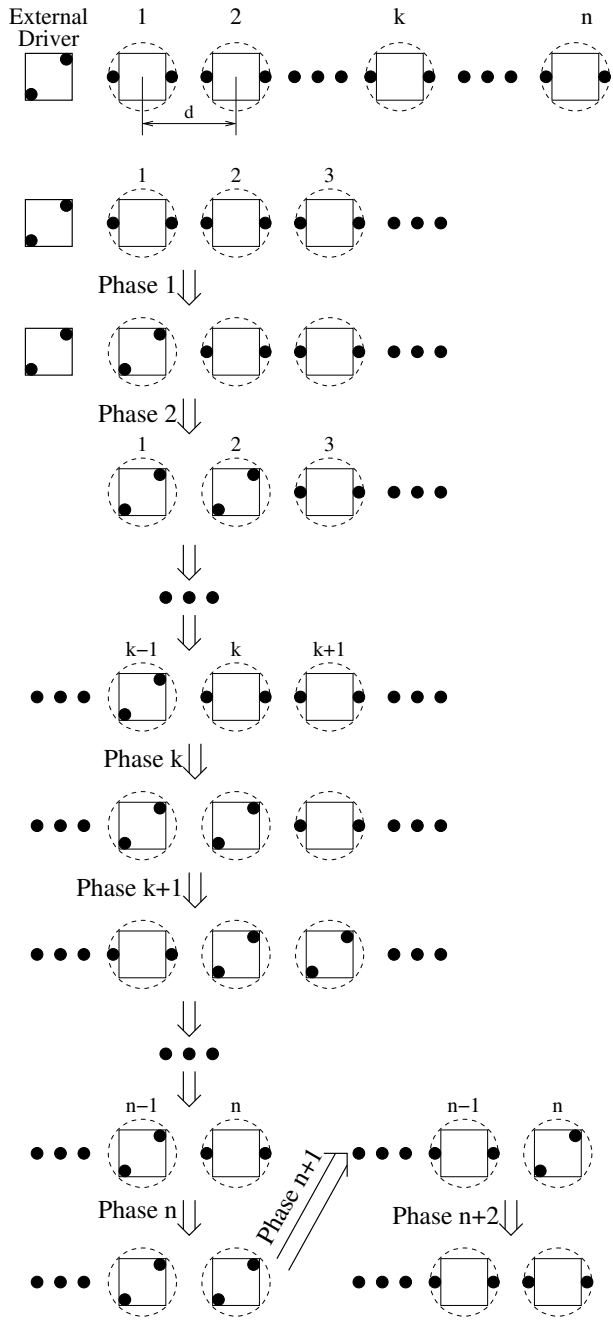


Figure B.2 Shift Register With One Cell per Stage (SR1)

Appendix C

Energy Dissipation Analysis of Circuit Units

X. Ma and F. Lombardi

The mechanical model can be applied to various QCA circuits to analyze the entropy change and energy dissipation. In the analysis, it is assumed that the parameters (q and a) of the device are selected such that the strength of the driver for each cell is sufficiently strong (as discussed previously in Section 11.3.1). For ease of presentation, only negative charged balls in the cell are presented in the figures of this section.

Shift register with multiple cells per stage (SR2): For a register whose stages consist of different numbers of cells (SR2), the non-dissipation feature also applies. When the kth stage is in the SWITCH phase, its cells are driven by the $(k-1)$th stage. When it is in the RELEASE phase, its cells are driven by the $(k+1)$th stage. So, as in SR1, the first $n-1$ stages in a n stage shift register work reversibly. If SR2 does not drive other circuits, each cell in its last stage will dissipate k_BT energy. If it drives other circuits, the entire SR2 works reversibly and does not dissipate any energy.

Fanout Circuit: For a fanout (Figure C.1), every cell inside the circuit is operated reversibly, too. However, there are two output cells, and they have no interaction with each other. So, if they are in the RELEASE phase without driving a subsequent circuit, the dissipation is $2k_BT$, twice as much as the dissipation of a single cell. If both outputs transfer information to subsequent circuits and are in the RELEASE phase while driving these circuits, then the fanout circuit is reversible. As a result, the fanout-then-erase circuit of Figure C.1 is reversible. The eraser

(cells 5 to 8) does not destroy information. Its inputs come from the fanout and can only take "00" or "11" as values. So, any input combination carries only one bit of information. The cell that erases two copies of information into one, operates as discussed in Section 11.3.1. It is reversible because it has the same polarization driver in the SWITCH and RELEASE phases. The driver strengths in the two phases are different, $E_{p1} = 2E_d, E_{p2} = E_d$, so $E_{p1} - E_{p2} = E_d$ will flow into the clocking unit. This conclusion can be extended to the n-to-1 reversible erasure, i.e., $(n-1)E_d$ energy flows into the clocking unit.

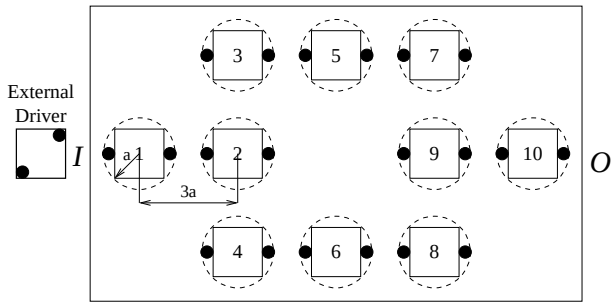


Figure C.1 Fanout and Reversible Eraser

The analysis above shows that the fanout structure by itself does not necessarily result in energy dissipation. In comparison with the normal connection (shift register), the increase of dissipation is associated with the erasure of an extra output cell. This conclusion holds also for the generic case of a one-to-n fanout circuit ($n \geq 2$).

Inverter: From the steady state energy calculation, it has been shown that the two input cells of the three-cell inverter are in the RELEASE phase under a same-polarization driver. So, only the output cell in the three-cell inverter dissipates an energy of $k_B T$ when released and with no transfer of information to a subsequent circuit. If the three-cell inverter connects to another circuit, then the output cell operates reversibly, too.

The inverter in Figure C.2 consists of a 1-to-2 fanout circuit and a three-cell inverter. So, it is also reversible.

Majority Voter: In the MV, if the inputs are 111 or 000, then no free expansion or damping will occur. The energy change is the same as the reversible erasure in the previous analysis. The voter cell erases 2 bit information reversibly and $2k_B T$ of energy goes into the clocking unit. If the input values are one of the remaining

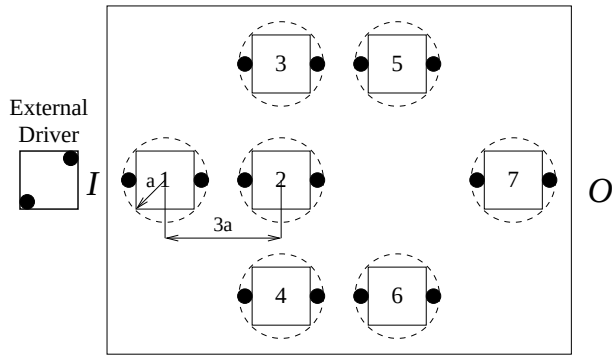
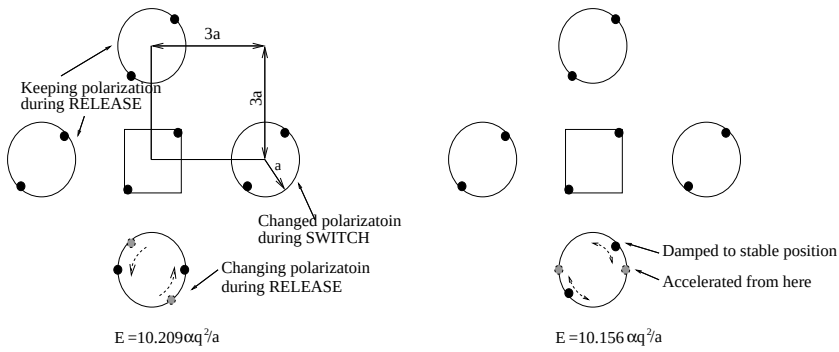


Figure C.2 One-input One-output Inverter as One-to-two Fanout and Three-cell Inverter

six possible combinations, then the input cell with the minority input will dissipate $k_BT + E_d$ energy when released under an opposite polarization driving condition (as shown in Figure C.3). E_d must be at least k_BT to ensure that the model operates reliably. Overall, at least $2k_BT$ energy is dissipated during the RELEASE phase.

So, there is 25% probability that the MV gets equal valued inputs and erases two bits of information reversibly. This does not dissipate energy directly, but an energy of $2k_BT$ goes into the clocking unit and will be finally dissipated into the environment to keep a stable clocking. There is 75% probability that the MV dissipates an energy of $2k_BT$ into the environment. Hence, the MV dissipates $2k_BT$ heat on average; due to loss, each input combination contains $-k\ \log_2 \frac{1}{8}$ of information, and each output only contains $-k\ \log_2 \frac{1}{2}$ of information. Therefore, two bits of information are destroyed in the MV and at least $2k_BT$ of heat must be dissipated. This dissipation is a lower bound imposed by logical irreversibility and it is in agreement with the expected dissipation as found from the above calculation.



(a) Before damping (b) After damping

Figure C.3 Damping in Majority Voter causes dissipation

About the Authors

Fabrizio Lombardi graduated in 1977 from the University of Essex (UK) with a B.Sc. (Hons.) in Electronic Engineering. In 1977 he joined the Microwave Research Unit at University College London, where he received a Master's degree in Microwaves and Modern Optics (1978), the Diploma in Microwave Engineering (1978), and a Ph.D. from the University of London (1982).

He is currently the holder of the International Test Conference (ITC) Endowed Chair Professorship at Northeastern University, Boston. At the same Institution from 1998-2004 he served as Chair of the Department of Electrical and Computer Engineering. He was a faculty member at Texas Tech University, the University of Colorado-Boulder and Texas A&M University.

Dr. Lombardi has received many professional awards: the Visiting Fellowship at the British Columbia Advanced System Institute, University of Victoria, Canada (1988), twice the Texas Experimental Engineering Station Research Fellowship (1991-1992, 1997-1998) the Halliburton Professorship (1995), the Outstanding Engineering Research Award at Northeastern University (2004), and an International Research Award from the Ministry of Science and Education of Japan (1993-1999). Dr. Lombardi was the recipient of the 1985/86 Research Initiation Award from the IEEE/Engineering Foundation and a Silver Quill Award from Motorola-Austin (1996).

Since 2000, Dr. Lombardi has been an Associate Editor of the *IEEE Design and Test* Magazine. He also serves as the Chair of the Committee on "Nanotechnology Devices and Systems" of the Test Technology Technical Council of the IEEE (2003 -). In the past, Dr. Lombardi was an associate editor (1996-2000) and the Associate Editor-in-Chief (2000-2006) of *IEEE Transactions on Computers* and twice a Distinguished Visitor of the IEEE-CS (1990-1993 and 2001-2004). Since January 1, 2007, he is the editor-in-chief of the *IEEE Transactions on Computers*.

Dr. Lombardi has been involved in organizing many international symposia, conferences and workshops sponsored by professional organizations as well as guest editor of Special Issues in archival journals and magazines such as *IEEE Transactions on Computers*, *IEEE Transactions on Instrumentation and Measurement*, the *IEEE Micro* Magazine and the *IEEE Design & Test* Magazine. He is the Founding General Chair of the IEEE Symposium on Network Computing and Applications.

His research interests are testing and design of digital systems, bio and nano computing, emerging technologies, defect tolerance and CAD VLSI. He has extensively published in these areas and coauthored/edited seven books.

Jing Huang received a B.S. degree in electronics engineering from Fudan University, Shanghai, China in 2001. She worked in the Computer Aided Test Lab in the Electronics Engineering Department, Fudan University, as a research assistant from 1999 to 2001. She received an M.S. degree in Electrical Engineering and a Ph.D. degree in Computer Engineering from the Electrical and Computer Engineering Department, Northeastern University, Boston, MA, in 2005 and 2007, respectively. She worked as research assistant at Northeastern University from 2003 to 2007; her research interests include testing, design for testability and fault tolerance of VLSI, reconfigurable systems and nanotechnologies. She is currently a design engineer at Sun Microsystems.

Mariam Momenzadeh received a Ph.D. degree in Computer Engineering from Northeastern University, Boston, in 2006. She received a M.Sc. degree in Computer Engineering and Science from University of Connecticut, Storrs, in 2003 and her B.Sc. degree in Electrical Engineering from Sharif University of Technology, Tehran, Iran, in 1999. Her research interests are testing, design for testability, ATE systems, defect and fault tolerance issues in digital systems and nano technologies, distributed and parallel computing, and fault-tolerant parallel algorithms.

Marco Ottavi received a Laurea degree in electronic engineering from the University of Rome "La Sapienza", Rome, Italy, in 1999 and a Ph.D. degree in microelectronic and telecommunications engineering from the University of Rome "Tor Vergata", Rome, in 2004. In 2000, he was with ULISSE Consortium, Rome, as a Design Engineer of digital systems for space applications. In 2003 he was a Visiting Research Assistant with the Electrical and Computer Engineering Department at Northeastern University, Boston, MA. Since 2004, he has been a Postdoctoral Research Associate at Northeastern University and during 2006 he was Visiting Research Scholar at Sandia National Laboratories in Albuquerque, NM. His research interests include yield and reliability modeling, fault-tolerant architectures, and online testing and design of nanoscale circuits and systems.

Vamsi Vankamamidi graduated with a B.S. degree in computer engineering from University of Mumbai, India, in 2000 and an M.S. degree in electrical engineering and computer science from University of Toledo, OH, in 2001. He is

currently working towards a Ph.D. degree in computer engineering at Northeastern University, Boston, MA. As part of his dissertation, he is working on quantum-dot cellular automata (QCA), a nanoscale device architecture to supersede conventional silicon- based technology. His research interests include the design of nanoscale circuits and systems, electronic design automation, defect tolerance and reliability.

Xiaojun Ma received a B.S. degree in Electronic Engineering (2001) and M.S. degree in Microelectronics (2004) from Fudan University, China. In 2004, he joined the Electrical and Computer Engineering Department of Northeastern University, Boston, MA. Since then, he has been studying as a Ph.D candidate. His current research interests are bio/nano computing, emerging technologies, reversible computing, defect tolerance and CAT/CAD.

Luca Schiano received a Laurea degree cum laude in electronic engineering from the University of Bologna, Italy, in 2001, and his Ph.D. degree in computer engineering from Northeastern University, Boston, MA, in 2004. He is currently a senior design engineer with Advanced Micro Devices (AMD).

His research interests vary from IC testing, ATPG, micro-processor testing, test data compression and reliability. to nanotechnology. Dr. Schiano has published more than 20 papers in international journals and conferences including *IEEE Transactions on Reliability*, *IEEE Transactions on Instrumentation and Measurement*, *IEEE Transactions on Nanotechnology*, *IEEE Design and Test Conference in Europe*, *IEEE Symposium on Defect and Fault Tolerance in VLSI Systems* and *IEEE Instrumentation and Measurement Technology Conference*.